Standardization and quality assurance of advanced MR-based protocols in radiology

Robin Antony Birkeland Bugge

A thesis presented for the degree of
Master in Physics

Department of Physics
University of Oslo
June 2014

Foreword

This thesis was written at the Intervention Center at Oslo University Hospital during the time period of august 2013 and june 2014.

I would like to thank my three supervisors: Atle Bjørnerud, Wibeke Nordhøy and Øystein Bech Gadmar for all their support and engagement in every aspect of the thesis. Also, thanks to Tone Elise Døli Orheim and Magne Mørk Kleppestø for assisting me with experiments and providing a lot of good insight.

A big thank you to all students and employees at the biophysics research group at UiO for providing a great work and social environment throughout my masters degree. And thank you to my patient girlfriend Susanne who has been there and supported me through all my work.

Oslo, Norway, May 2014
Robin Antony Birkeland Bugge

Abstract

Standardization and quality assurance (QA) are essential both for routine magnetic resonance imaging (MRI) clinic and for research. Yet, the extent to which QA is performed varies significantly. Several scanner vendors offer to do yearly and quarterly QA, but then utilizing their own software into which the end user rarely gets insight. The Norwegian Radiation Protection Autority made recommendations in 2005 that urged all clinics and centers using MRI to have at their disposal a quality assurance methodology that is independent of the MRI vendor. The methods used in yearly quality assurance tests performed by physicists are often varying and manually intensive. One consequence of this is that results vary to such a degree that cross-vendor comparisons becomes difficult. This is especially a challenge in larger institutions with hardware from several vendors. Several of the methods used contain sources of error based on human interactions.

In this master thesis the main goal has been to remedy this situation by creating an easy to use software package that reduces the amount of manual effort and makes it more feasible for trained personnel to keep track of MRI system performance over time. This software will contain a graphical user interface (GUI) for interacting with images and specifying analysis-specific options. The main focus has been on quality assessment in terms of geometric distortions, signal to noise ratio (SNR), signal uniformity and temporal signal drift in functional MRI (fMRI). All code has been written in Matlab (R2012b) with the Image Processing Toolbox.

A major goal of the project was to automate the currently time consuming task of measuring geometric distortion based on standardized phantom images. Through the use of pattern recognition methods, phantom markers could be automatically identified thereby facilitating automated analysis of distortion in a reproducible manner with equal accuracy compared to manual analysis. The use of Hough algorithms for detecting circular objects in the structured module of phantoms in the program, will reduce subjective measurements and human error, as well as lead to better regularity and consistency in QA analysis and reports. The program follows NEMA and IEC procedures and recommendations [1] [5]. In order to compare the results between manual and automatic calculations of the geometric distortion, artificial MRI images with known distortion percentages were constructed. The deviation between the values calculated by the software and the actual values was calculated in over 100 images based on the root mean square for indi-

vidually measured points. A similar test was done to compare with human manual measurements. In this case ANOVA tables and t-tests showed no difference between manual and automated anlyses, but mostly due to a few manual anomalies that may or may not be common.

The signal to noise ratio measurements were compared with previously reported measurements that used a different program written in IDL. There were some discrepancies in the results, which in some cases could go as high as 10 % . The reason for this is thought to be the different methods in object detection, which causes the region of interest to cover slightly different areas. For non-homogenous images this will affect the SNR.

In general the QA program operates without any major issues and performs at least as well as a trained MR phycisist. It makes the QA procedures more seamless and substantially faster, and removes the need for cumbersome manual input both in the measurement and analysis phase of the procedures.

Contents

Abbreviations and acronyms

BOQA	Book of Methods in MR QA
CHT	Circular Hough Transform
EPI	Echo planar imaging
fMRI	Functional magnetic resonance imaging
GD	Geometric distortion
MRI	Magnetic resonance imaging
OUS	Oslo Universitetssykehus (Oslo University Hospital)
QA	Quality assurance
QAP	Quality assurance program, developed for this thesis
ROI	Region of interest
SD	Standard deviation
SE	Spin echo
SNR	Signal to noise ratio
TE	Echo time
TR	Repetition time

Chapter 1

Introduction

1.1 Background

The physicists in the clinic carry out different tasks related to their field of expertise, some of which are routine based performance services that relate the need for calibration of various systems. In medical technology these tasks are referred to as quality assurance, QA for short, and is an integrated part of the clinical work environment. The Norwegian Radiation Protection Authority (NRPA) have since 2005 required all facilities utilizing clinical MRI to have an available qualified phycisist and to have an independent system for quality assurance [12].

This thesis has focused on establishing automated quality assurance routines that will function as auxilliary and independant methods within a subset of the regular QA-routines. A standard QA procedure often includes measurement of SNR, signal uniformity, contrast resolution, ghosting level and geometric distortion, which is part of the routine tasks for the MR physicists, whereas the aim of this thesis was to develop and implement robust and automatic software for some of the most important QA methods in MRI. These are, in addition to basic image quality methods such as spatial SNR and signal uniformity, geometric distortion and temporal signal stability methods.

The field of diagnostic radiology, and especially MRI, is one that requires good QA routines that are carried out on a regular basis. Currently, at the OUS (Oslo University Hospital) the MR physicists are carrying out a standardized QA procedure once a year on each and every MRI system in the hospitals in the South-Eastern Health Region of Norway. In addition, the

various manufacturers perform their own QA routines every 3 to 6 months.

Automated procedures are generally preferred if sufficiently accurate since they eliminate user bias and potentially save time. One example is geometric distortion, where manual measurements are subject to user bias, which is dependent on gauging the positions and lengths in a geometric structure. There is also time saved from drastically reducing the number of steps that a human must intervene in the process.

As already mentioned, most suppliers of MRI equipment provide QA at regular intervals as part of their services that differ in content and frequency amongst vendors. Despite of this it is of great importance for both research and the clinical work to have independent methods for testing. This is not only useful as a means of self-assurance, but is essential for any center that needs to ensure that their MR machine is stable and produce high quality images, and for early discovery of various errors that are not yet grown large enough to become visually identifiable in the images. There are too many things that can go wrong with a MRI system to give an extensive overview of various sources of error in this thesis. That is everything from malfunctioning gradient coils to a patient dropping a metallic or magnetic coin in the machine interior. These are situations that can occur in the time between QA runnings and needs to be rectified, which are possible to detect using the appropriate on-site QA method.

1.2 Goals of the thesis

The goals of this thesis is to: Reduce user bias in QA assessment of multi-vendor MRI data through the development of a validated semi-automated QA software tool.

In order to achieve this the following specifications and part-goals were utilized:

- Investigating current methods for performing quality assurance in MRI.
- Creating new methods or altering existing methods to better suit the needs of this project.
- Incorporate these methods in a Matlab based semi-automatic QA program that will function as a supplement to the current quality assurance procedures.

- The QA program must be vendor independent, in such a manner that it can be used to analyze images from any MR machine.

One of the main concerns that was introduced to the author from the beginning, was that the QA procedures took a long time to perform: that they where cumbersome due to the need for extensive manual interaction; and that they where subjective since estimates of object positions had to be made from visual interpretations alone. One of the key aspects of the project was therefore to automate the process, and make QA easier and more accessible to everyone involved in MRI. The automatization of the QA analysis reduces the amount of necessary user interactions, thus shifting much of the workload onto the computer. It must be mentioned though, that this removes a lot of the flexibility of the method and can obscure the end result unless one has good insight and experience with this software. There has been put a lot of effort in trying to make a user-friendly graphical user interface and a readable output format. This program will be a useful tool that can be developed further or serve as an idea for future projects of a similar nature.

Chapter 2

Theory

2.1 MRI theory

This section addresses some basic principles of MRI. This is only meant to be a brief introduction to MRI, where more comprehensive theory can be found in most books on the subject. This section uses information found in references [6] and [8].

Nuclear Magnetic Resonance (NMR, most often shortened to MR) is the interaction between nuclear particles with a magnetic moment, such as protons, and an external magnetic field. Hydrogen nuklei only consist of a single proton that have spin 1/2 and are heavily abundant in organic tissue, which largely consists of water. This makes it very suitable for clinical MRI. The total magnetic moment formed by the spins will rotate, or "precess", around the main magnetic field, B_0, with an angular frequency called the Larmor frequency, ω_0:

$$\omega_0 = \gamma B_0 \tag{2.1}$$

,where γ is the gyromagnetic ratio, characteristic for the nucleus considered. That is, $2 \cdot \pi \cdot 42.6 \cdot 10^6$ Hz/T for protons.

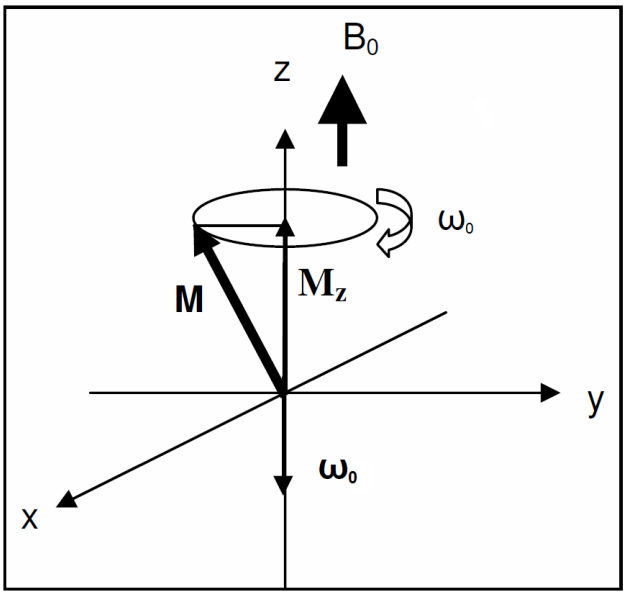


Figure 2.1: Illustration of the principle of precession. The magnetization vector M is precessing around the z-axis with the Larmor frequency ω_0. Image is a modified version of an image borrowed with courtecy of reference [6].

2.1.1 Excitation and relaxation

The basic principle behind MRI is proton density measurement using oscillating magnetic fields in the radio frequency spectrum. There are two phenomena that enables this, precession and excitation.

By applying an oscillating B_1 field with a frequency of ω_0 in the form of a pulse and with an energy equal to $\gamma\hbar B_0$, an energy transition in the spin states of the protons can be induced. This is referred to as excitation. The proton spin is flipped from one of two states to the other, parallel or anti-parallel to the B_0 field and is the result of the proton flipping from the low to the high energy state. The sum of all the small magnetic moments builds a net magnetic moment, or macroscopic magnetization, M. In its non-excited state, this magnetization is directed along the B_0 field in the z-direction. However, after an excitation pulse there will be a magnetization component in the xy-plane, with an amplitude that depends on the magnitude and duration of the RF-pulse.

Excitation can be described as an extension to the Bloch equation describing both precession and excitation. It shows the relationship between the macroscopic magnetization M,the magnetic flux density B_0 and the oscillat-

ing magnetic field B_1.

$$\frac{dM}{dt} = \gamma(M \times B) = \gamma M \times (B_0 + B_1) \tag{2.2}$$

The principle of induction enables recording of electrical signals from the macroscopic magnetization in the xy-plane, named M_{xy}. After an excitation pulse this signal gradually decays with time, this is called relaxation. Relaxation is the loss of energy from an exited system due to proton interactions. Dipole-dipole and chemical shift interactions cause random magnetic fields near the Larmor frequency, and the resulting energy transition causes recovery of the z-component of the magnetization, M_z. This is called T1 relaxation, spin-lattice relaxation or longitudinal relaxation. T2 relaxation describe transversal magnetization decay. It is also known as spin-spin relaxation. T2 relaxation is due to field inhomogeneities on the molecular level, created by the sum of several shielding effects and macroscopic inhomogeneities. Figure 2.2 shows T1, T2 and T2* relaxation. The latter is defined as:

$$\frac{1}{T2*} = \frac{1}{T2} + \gamma \Delta B_0 \tag{2.3}$$

Here ΔB_0 is the deviation of magnetic flux density in the static magnetic field B_0.

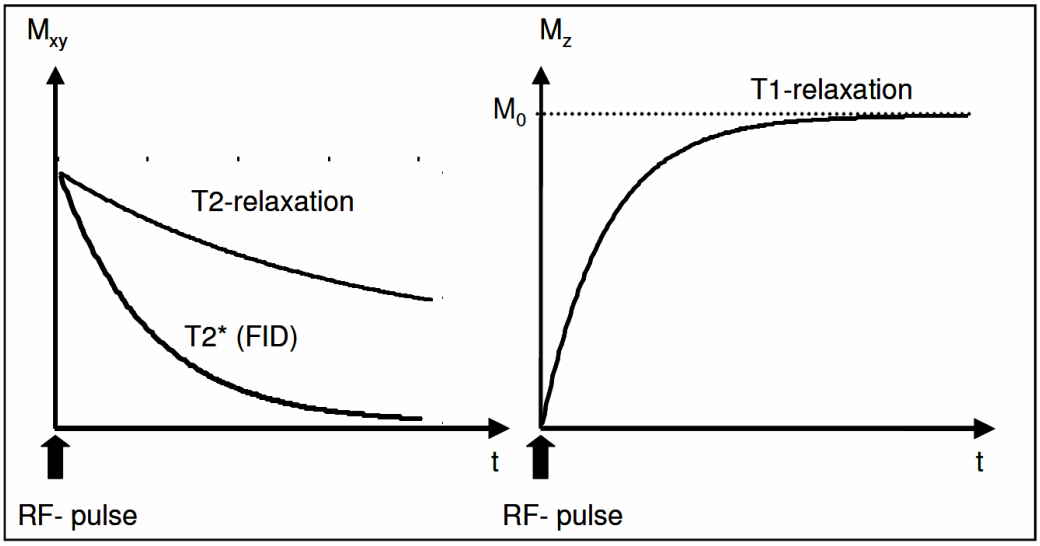


Figure 2.2: Example plots of different kinds of relaxation: T2 and T2* (left) relaxation, which are called transverse relaxation types. T1 relaxation (right) which is called longitudinal relaxation. Image borrowed with courtecy of reference [6].

2.1.2 Slice selection

When excited, all the spins in the object will precess at the same Larmor frequency. Spin excitation as described above does not provide spatial selection of the excitation process. The magnetic field strength has to be position dependent so that a predefined section (slice) of the object can be excited, it is then possible to build the image from a series of these slices.

This is done by adding a magnetic field gradient along the z-axis. The position dependent field strength is then defined as:

$$B_z(t) = B_0 + G(t)r \tag{2.4}$$

Where G is the magnetic gradient field strength vector in units mT/m, and r is a position vector.

2.1.3 k-space and sampling

In order to represent the spatial frequencies of the image signal, the k-space is the most common way of demonstrating pulse sequence signal acquisition.

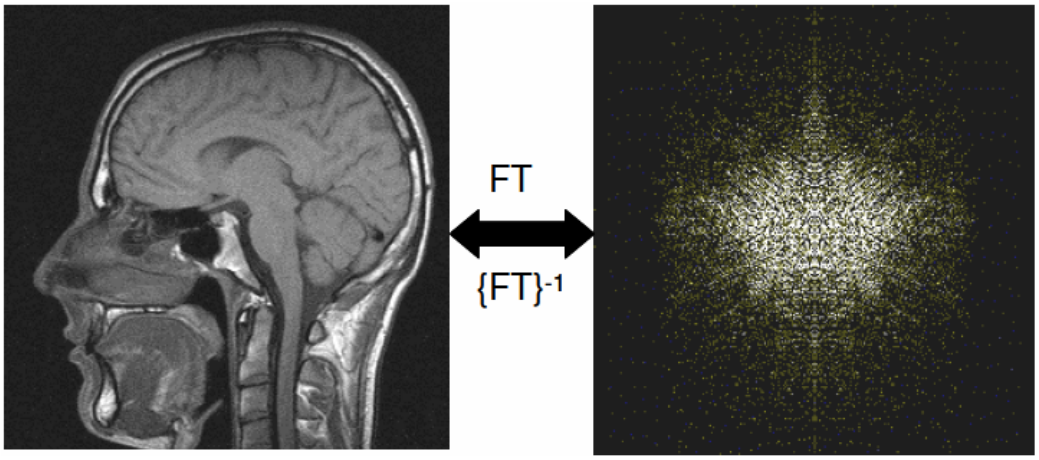


Figure 2.3: An MR image and its fourier transform equivalent.

Originally introduced as a concept of the spin density operator, k-space is the fourier transform of the MR image. Figure 2.3 shows both an MR image and the fourier transform of that same image which is known as the k-space image. The x-axis of the k-space image is the frequency and so called read-out direction, and it is this axis that is sampled as a line for each step through the y-axis, known as the phase-encode direction.

2.1.4 RF-pulse sequences

Gradient Echo sequences

The gradient echo sequence (GRE) is a pulse sequence where one line (frequency direction, x-direction) is sampled with each rf-pulse.

Figure 2.4 shows the GRE sequence step by step. The rf-pulse initiates the excitation at a slice given by the G_z gradient, and is followed by positioning gradients in the x and y direction. The initial sampling position is achieved by combining an x-gradient and y-gradient that sums according to equation 2.8 shown later in this section, this is referred to as the preperation gradient. This is followed by a read-out gradient G_x that creates a frequency modulation of the signal as it is sampled [6].

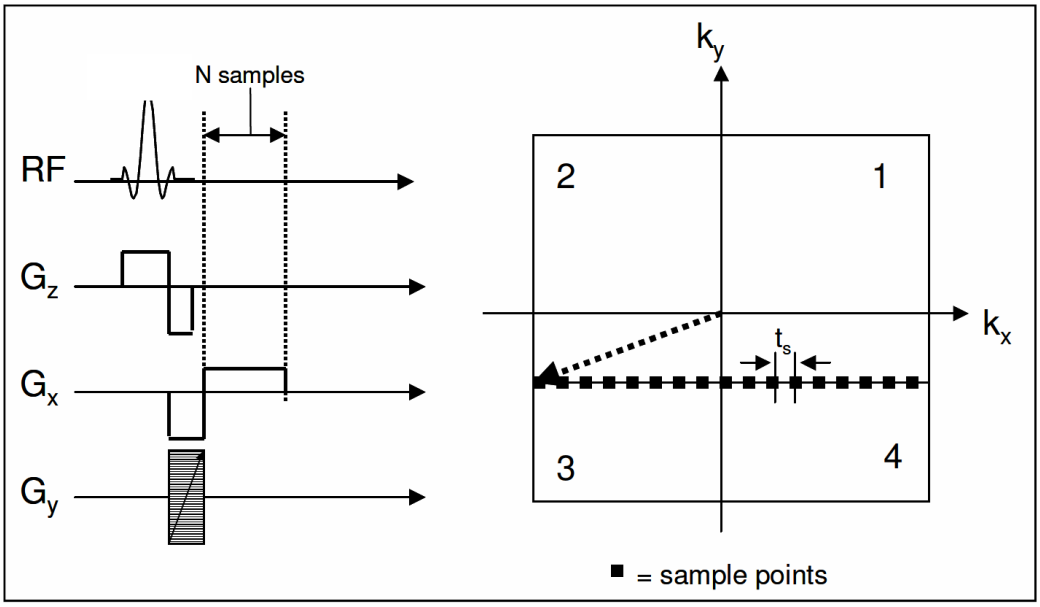


Figure 2.4: Illustration of the gradient echo (GRE) sequence. Left side of the image shows the sequence of rf-pulse and magnetic gradient fields. The right hand side shows the sampling of k-space. Image borrowed with courtecy of reference [6].

Echo Planar Imaging sequences

Echo Planar imaging (EPI) is the fastest acquisition method in MRI. One single image can be acquired in about 100 ms, and it is therefore used in functional MRI where this speed is essential for monitoring systems in rapid change. The main drawback of this high speed is increased image artifact sensitivity. EPI is similar to GRE, but differs in that EPI samples several lines in k-space for each rf-pulse. A "single shot acquisition" refers to sampling the entire k-space and produce a complete image with a single rf-pulse.

In figure 2.5 each of the G_y fields moves us one step down to the next k-space line to be sampled. Figure 2.6 shows us the pathway taken through the k-space during the acquisition of an EPI image.

Equation 2.5 shows that when the time t after excitation becomes smaller, both T2* and δB related artifacts becomes bigger.

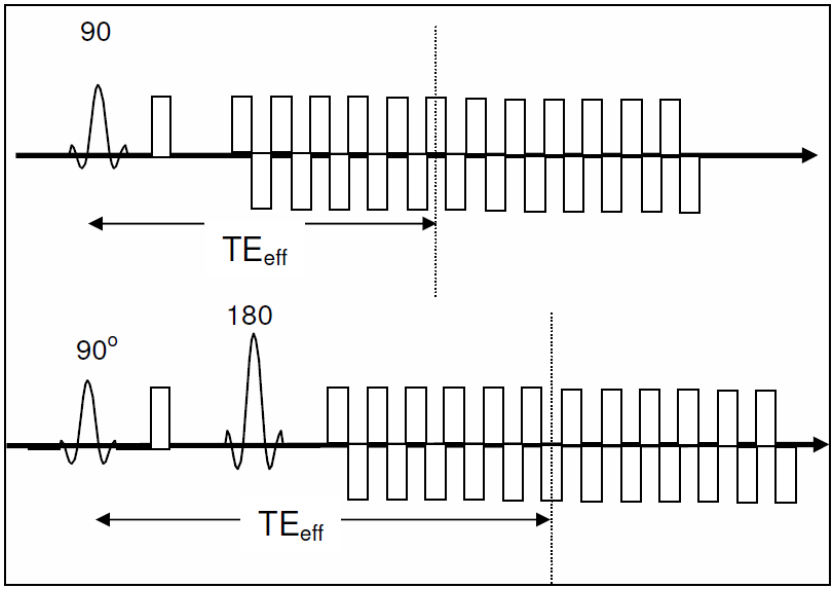


Figure 2.5: Illustration of Echo Planar Imaging (EPI) sequences. The top of the figure shows FID-EPI, and the bottom of the figure shows SE-EPI. Image borrowed with courtecy of reference [6].

2.2 MRI system overview

The MRI system consists of several components that are vulnerable to variations in accuracy over time. The signal from magnetic resonance imaging is dependent on many variables related to the individual components of the system and on the imaging sequence. The spin density (ρ(x,y)) can be related to the total transverse magnetization, T2* relaxation, and field inhomogeneities through a fourier transformation [6]:

$$\rho(x,y) = \frac{1}{2\pi}\int_{k_x}\int_{k_y} M_T(k_x,k_y)exp(j(k_x x+k_y y))exp(-t/T2^*)exp(-j\delta B(x,y)t)dk_x dk_y \tag{2.5}$$

ρ is spin density at a certain location (x,y). $\mathbf{M_T}$ is the total transverse magnetization vector. t is time after excitation, and T2* is the transversal relaxation time related to T2 decay and bulk inhomogeneities. δB is the B_0 inhomogeneity. According to equation 2.5 field inhomogeneities can lead to signal loss (through enhanced T2* relaxation) and signal displacement and consequent geometric distortion through the δB term.

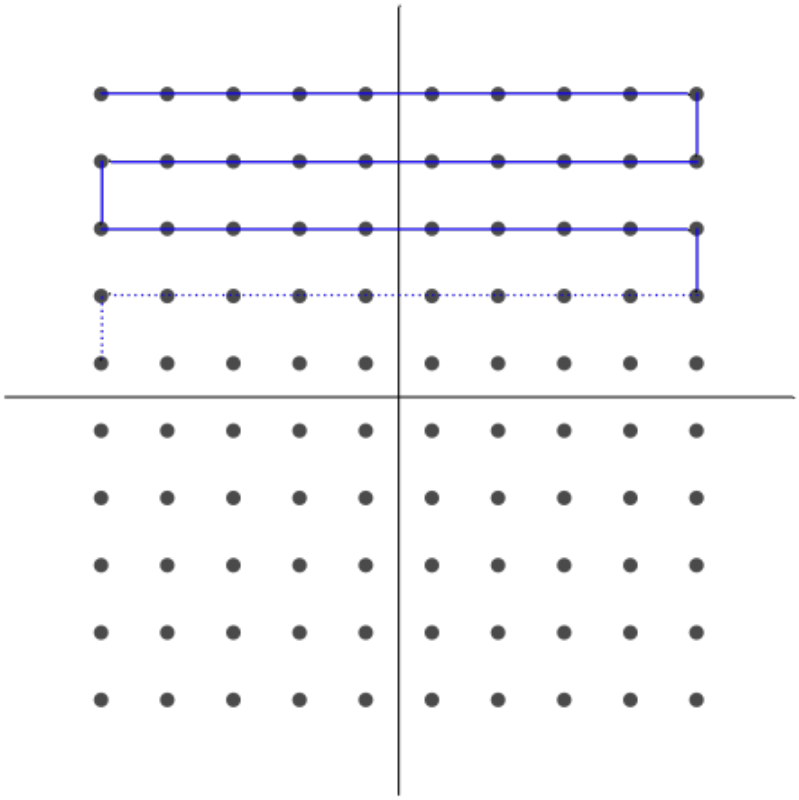


Figure 2.6: Path of sampling in the k-space for Echo Planar imaging (EPI).

2.2.1 System components

Though there are several methods of creating a magnetic field strong enough for the purpose of MRI, all the machines used in this thesis utilizes a liquid helium cooled (4.3 K) electromagnet in order to achieve high quality high density flux densities (1.5 T - 3 T) with high homogenous field spheres. The cryostat magnet contains a wire made from a niobium titanium alloy inserted in copper that becomes superconductive when colder than 12 K [8]. This enables the wire to carry a high density current without any resistance and thus without a source of voltage. Magnetic field strength is proportional to the current I through the wire and number of turns per meter N:

$$B_0 = \mu_0 I \cdot N \tag{2.6}$$

However due to non-ohmic losses and helium evaporation there is a need for periodic retuning of the magnet current and refilling of helium. The magnet

current changes during recharging according to:

$$U = L\frac{dI}{dt} \tag{2.7}$$

Here U is voltage of the power supply, L is the coil inductance and I is current.

The gradient coils are necessary for position encoding of the MR-signal. Unlike the coils that generate the homogenous B_0 field, the gradient coils consist of resistive, and not superconductive, materials. In principle there are three coils that generate a field in a specific combination of the x, y and z components. Where the B_0 field is directed along the z axis. This system can be described by the equation:

$$B_{G,z} = \frac{dB_x}{dx}x + \frac{dB_y}{dy}y + \frac{dB_z}{dz}z = G_x x + G_y y + G_z z \tag{2.8}$$

These gradient coils (G_x,G_y,G_z) in combination with the radiofrequency pulses emitted by the transmit coils, enables spatial encoding of the NMR signals and makes it possible to do slice selective excitation of an object or tissue. Figure 2.7 shows an overview of the most important components in an MRI machine. Inside a shielded room, the patient lies on a moveable table surrounded by magnet cryostats and several coils used for transmission and recieving of signals.

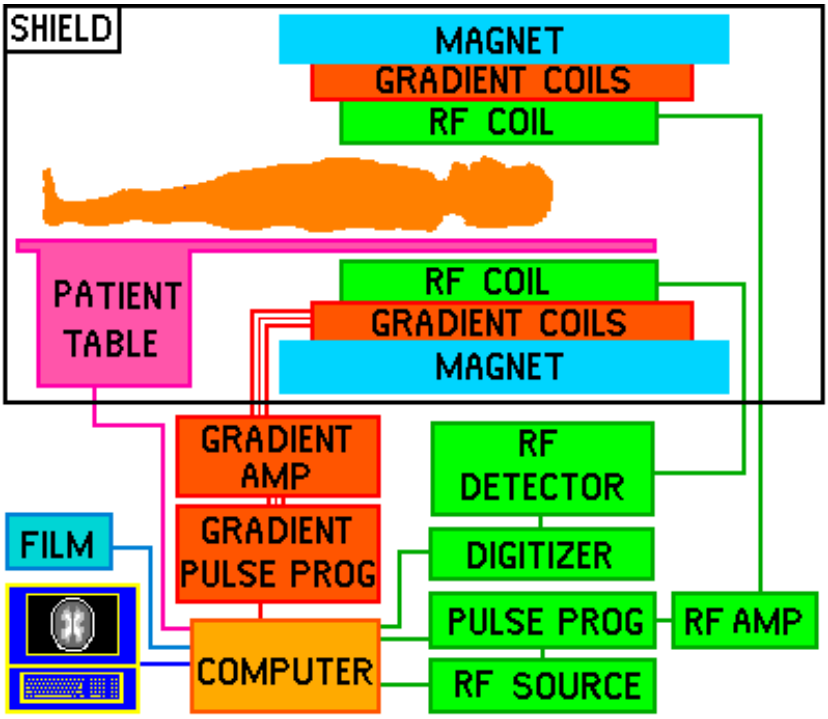


Figure 2.7: MRI hardware overview. The figure illustrates the major components of an MRI system. It consists of two magnet cryostats that supply the main magnetic field B_0. A transmit coil, also called a body coil, is used to exite the object of interest. A recieve coil picks up on the changing total magnetization through induction.

2.3 Sources of error

2.3.1 Sources of geometric distortion and inhomogeneities

Geometric distortion can occur as a result of inhomogeneities in the main magnetic field B_0 and in the gradient coil system. These inhomogeneities appear in all MRI systems as a design issue, and are therefore to a certain degree known. To some extent the inherent inhomogeneities of the main magnetic field of the MR system are compensated for using mathematical correction algorithms. However, over time, any given machine may be subject to changes in these calibrated values so that the inhomogeneity compensation will become out of tune with the actual field.

Susceptibility differences and eddy currents within the object of interest will

also contribute to geometric distortion. As can be seen from equation 2.9 both changes in susceptibility and static magnetic field will contribute to an apparent shift in pixel location.

$$\Delta x = \frac{\Delta \chi B_0}{G_x} \tag{2.9}$$

Δx is the shift in pixel position, Δ χ the shift in magnetic susceptibility, B_0 the static magnetic flux density and G_x the peak magnitude of the read-out gradient.

Geometric distortion in images should be taken seriously while it can affect the interpretation of images in both clinical settings and research.

2.3.2 Sources of drift in time series

Functional MR (fMRI) sequences like echo planar imaging (EPI) are very fast when compared to the GRE sequence. When it is utilized it generates heat both in the coils and phantom or subject of interest (though the latter is rare for most sequences). This heat can alter constants such as relaxation times in the subject and the conductivity properties of the coils, thus changing the output signal value. For hardware components such as gradient coils, the signal-amplitude variations (and possible geometric distortions) are fairly common in fMRI. This change in signal can be a problem where you have a series of images taken consecutively. Signal drift, is when the signal changes from one image to the next due to heating in the system. It is important to have the highest amount of signal possible, not meeting a specific limit. The exact signal value is not important [7]. However, it is important to gain internal consistency when attempting to observe a system that undergoes change. Apparent changes in the signal value can make it seem as if an actual change is happening in the imaging system.

2.4 General QA methods

Three of the general quality assurance methods that are commonly employed in MRI are: Spatial signal to noise ratio (sSNR, usually just written SNR), uniformity which is also called homogeneity, and geometric distortion (GD). The theory in this section is found in references [1], [2], [3], [4], [5] and [19].

2.4.1 International guidelines and standards

The general QA methods currently used at OUS follows NEMA and IEC guidelines for QA image recording and analyses. These guidelines are widely accepted as the standards in quality assurance in MRI.

NEMA

NEMA (National Electrical Manufacturers Association) is a membership based corporation with a north american focus. NEMA has about 400 membership companies wordlwide that produce different kinds of electrical equipment including but not excluded to energy generation, transmission, distribution and end use. A part of this is medical imaging. NEMA explains their goals on their website:

"NEMA provides a forum for the development of technical standards that are in the best interests of the industry and users, advocacy of industry policies on legislative and regulatory matters, and collection, analysis, and dissemination of industry data. In addition to its headquarters in Rosslyn, Virginia, NEMA also has offices in Beijing and Mexico City."

IEC

IEC (International Electrotechnical Commission) is an international organization that publishes standardization works based on pooled experts from a variety of fields, such as industry, commerce and academia. IEC states on their website:

"Founded in 1906, the IEC (International Electrotechnical Commission) is the world's leading organization for the preparation and publication of International Standards for all electrical, electronic and related technologies. These are known collectively as "electrotechnology".

IEC provides a platform to companies, industries and governments for meeting, discussing and developing the International Standards they require."

2.4.2 Spatial SNR

The signal to noise ratio (SNR) is defined as the relationship between a signal and the noise in an image. How this is calculated in practice varies, whereas in NEMA these are divided into two types of SNR that are named SNR1 and SNR2

For SNR1 and SNR2 one makes a difference image based on two seperate images taken with identical acquisition methods without any adjustments or calibrations between scans [3]. Since the magnitude image has a Rician noise distribution the difference subtraction is used for the noise calculation:

$$[Image3] = [Image1] - [Image2] \tag{2.10}$$

The standard deviation (SD) of the pixel values inside a ROI in the difference or subtraction image (Image3) is calculated using two different methods:

SNR1:

$$SD_1 = \frac{\Sigma_{i=1}^{n}\Sigma_{j=1}^{m_j}(V(i,j) - \overline{V})^2}{\Sigma_{i=1}^{n}(m_j) - 1} \tag{2.11}$$

SNR2:

$$SD_2 = \frac{\Sigma_{i=1}^{n}\Sigma_{j=1}^{m_j-1}(V(i,j+1) - V(i,j))^2}{2\Sigma_{i=1}^{n}(m_j - 1)} \tag{2.12}$$

In equation 2.11 and 2.12 V(i,j) is a pixel value from Image3, $\bar{V}$ is the average pixel value in Image3, i is an index that spans the read encode direction and j is an index that spans the phase encode direction.

Image noise is acquired by dividing the relevant SD by $\sqrt{2}$, which is used to correct for the use of a difference image. S is the average image signal within the ROI in Image3.

SNR is then defined as:

$$SNR = \frac{S}{Noise} = \frac{S}{SD \cdot \sqrt{2}} \tag{2.13}$$

SNR1 and SNR2 are spatial SNR methods, giving the signal to noise ratio for a single image from the difference between two images acquired under identical conditions. For both methods, SD is calculated from the difference image, but they differ in that the SD of SNR1 is subtracting the mean signal value from each pixel value (equation 2.11), whereas the SD of SNR2 is calculated by subtracting the neighboring pixel values from each other (equation 2.12). Therefore SNR2 is less sensitive to image inhomogeneities. These inhomogeneities can occur due to systematic changes between the two separately acquired images (image1 and image2).

2.4.3 Uniformity

The theoretical values and requirements used by the International Electrotechnical Commission (IEC) [5] have been applied for calculating uniformity.

In order to acquire a good image for testing homogeneity measurements a homogeneous phantom containing a liquid medium with T1-values comparable to human tissue is commonly used. The uniformity U is found by first calculating the average absolute deviation (AAD) inside the defined ROI:

$$AAD = \sum_{i=1}^{N} \frac{|S_i - S|}{N} \tag{2.14}$$

S_i is the individual pixel value inside the ROI, S is the mean value of all the pixels inside the ROI, and N is the total number of pixels inside the ROI.

$$U = 1 - \frac{AAD}{S} \tag{2.15}$$

The uniformity U is then given as a value between 0 and 1, where 1 is a perfectly uniform image.

2.4.4 Geometric distortion

Geometric distortion (GD) is the warping of an MR image due to off-resonance effects. This can be due to local variances in the nominal value of the static magnetic field B_0, non-linear gradients or because of sucseptibility differences within an object. GD will often be most prominent in the edges of an image where the static magnetic field more often experiences inhomogeneities, though susceptibility effects can affect the image anywhere where there is a transition between medias with different susceptibility.

Each machine vendor, e.g Philips and GE, deliver MRI systems with certain specifications for the maximum distortion one can accept in the outer most areas of the image. The unit of distortion can be given as percentage of difference between a measured and a given diameter of sphere volume. Though this is dependent on the particular method one is following.

Two methods of assessing GD will be considered, the NEMA and IEC methods.

In NEMA, the method for measuring GD is the length between each object and the center of the phantom and calculates the difference between this length and that of the known positions in the phantom. When GD is measured manually, only the outer most objects are taken into account, see figure 2.8. This is due to the fact that GD is rarely larger anywhere than in the outer parts of the image in such controlled situations as a phantom recording. In the automated procedure programmed in Matlab, all the objects in the GD module are taken into account due to negligible amount of extra time spent to calculate positions and lengths. Also, the object distance from isocenter is used to normalize distortion values and not letting it be subject to the variable results one can find in object detection algorithms.

IEC uses the concept of scaled geometric distortion and measures lengths between phantom center and a set of measure points in the peripheral parts

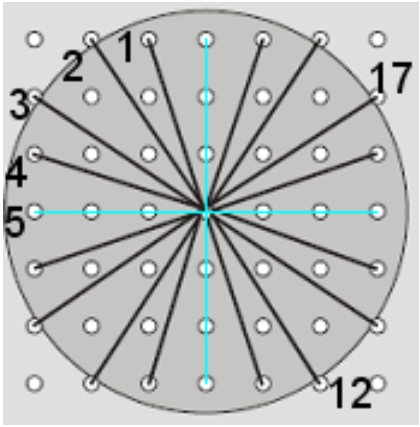


Figure 2.8: NEMA method for determining geometric distortion. Each line gives the distance from the center of the phantom to each object center. According to the book of methods in MR QA used at the OUS, the added turquoise lines are an addition to the suggested points of measurement from NEMA [19]

of the phantom. In the IEC report it is assumed that each point is originally at equal distance from the phantom center, and a scaled geometric distortion is defined as:

$$\delta = 1 - \frac{\bar{r}}{R} \tag{2.16}$$

Where $\bar{r}$ is the mean of the measured distances, R is the actual phantom radii. The phantom used in this thesis does not have equal distances between the outer most measure points. This is circumvented by normalizing the distances by varying the length R for each point.

Automated GD analysis based on CHT

Matlab provides a function called imfindcircles that utilises Circular Hough Transform (CHT) as a means for detecting circular objects in an image [13]. The CHT method utilized the geometry of circles and a pixel voting system known as accumulator array computation to make an estimation of circle radii and center position.

High gradient pixels are set to be tested as edge pixels, and so an accumulator array is made of pixels that are in a set radii from said pixel.

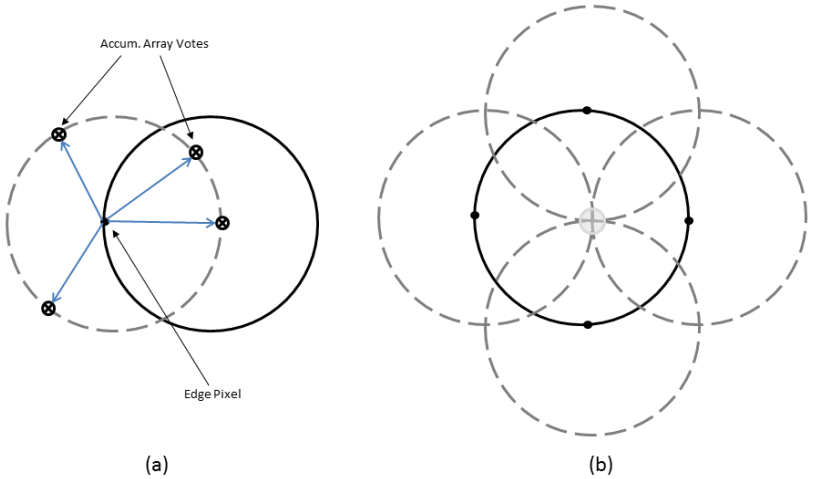


Figure 2.9: Circular Hough Transform accumulator array. The solid line circles are objects that are going to be detected. The dotted lines are arrays of circularly placed votes that detect pixel signal values over a threshold limit. Several of these arrays, as seen in b, will intercept at an approximate object center when the array radii is at a value that is found experimentally by incremental radii increases.

The second step in the CHT process is called center estimation. After generating several accumulator arrays using different edge pixels, a circle center can be estimated from the coinciding points of these arrays see figure 2.9. The final step is finding the radii of the circle which is the same as the radii of the estimation circles used.

2.5 QA methods for functional MRI

The methods and theory presented in this section does not adhere to any international standards, but is in accordance with published literature [7] [9] [15].

2.5.1 Temporal SNR

tSNR is a measure that is meant to describe how spatial SNR is changing over time. Temporal SNR is not commonly used in general QA, and therefore not a method of the local physicists in BOQA (Book of methods in MR QA), but

a QA measurement that is designed for dynamic or functional MRI methods, such as perfusion imaging (Dynamic susceptibility contrast, dynamic contrast enhanced MRI and arterial spin labeling) and blood oxygen level dependent MRI (BOLD). The tSNR is used for time series from fMRI and is estimated by creating two images representing pixel-wise mean intensity and temporal fluctuations in the time series, shortened tSMI (temporal Signal Mean Image) and tFNI (temporal Fluctuation Noise Image).

tSMI is defined as an image of equal dimensions to the images in a time series, where each pixel (x,y) value is the mean value of the pixels (x,y,:) in the series.

$$tSMI_{x,y} = \frac{\Sigma_{i=1}^{N}(S_{x,y,i})}{N} \tag{2.17}$$

Here $S_{x,y,i}$ is the signal value in position (x,y) in image i, and N is the number of images in the time series.

A new image SFNRI (Signal Fluctuation Noise Ratio Image) formed by the ratio between tSMI and tFNI is then defined as:

$$SFNRI = \frac{tSMI}{tFNI} \tag{2.18}$$

tFNI is the standard deviation of the signal value in the pixels (x,y,:). And the tSNR (temporal Signal to Noise Ratio) is then the mean value of the pixel values inside an assigned ROI in the SFNRI:

$$tSNR = \bar{S} \tag{2.19}$$

Here $\bar{S}$ is the mean value of all pixel values inside the ROI in the SFNRI.

2.5.2 Drift

Drift is signal intensity variations that can occur in fMRI-imaging when comparing two images in a time series. The baseline of the signal can change over time in response to heat development in the coils and system hardware.

Drift value D is given as a percentage of change in signal intensity divided by the mean signal accross several images. The values S_{max}, S_{min} and S_{mean}

are the maximum, minimum and mean values of a second-order polynomial fit.

$$D = 100 \cdot \frac{S_{max} - S_{min}}{S_{mean}} \tag{2.20}$$

Weisskoff EPI stability test

The Weisskoff stability test for EPI sequences is a comparison between two relative coefficients of variation: One theoretical value based on the geometry of a region of interest and a measured value. The theoretical coefficient of variation (CV) is given as:

$$CV = \frac{SD}{mean} \cdot 100 \tag{2.21}$$

A good Weisskoff plot is one where the measured values of CV follow the theoretical values until higher ROI radii. It is expected that CV will scale inversely with ROI area. If CV deviates from this theoretical CV when ROI size increases it is assumed to be due to low frequency vibrations due to some form of hardware instability [7]. The classification of what describes a good ROI radii is somewhat subjective, but is described in [7] as an ROI width of approximately 20 pixels, as seen in figure 2.10a. A bad Weisskoff plot is shown as one where the data deviate at ROI width of 2 pixels and stop declining at 6 pixels, as seen in figure 2.10b.

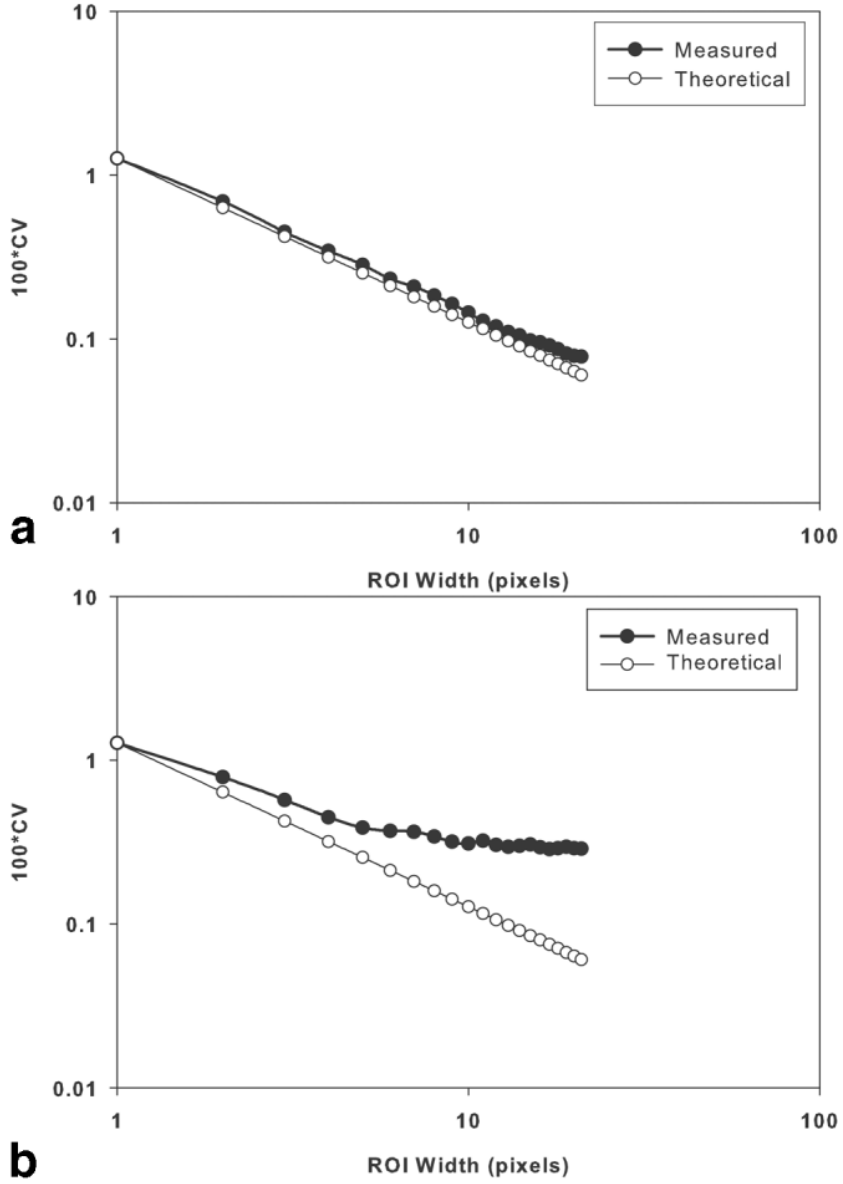


Figure 2.10: a and b are showing two examples of Weisskoff plots with different degree of overlap between measured and theoretical CV values. Where a is considered good overlap, while b is less good. Figure does not belong to author and is slightly modified (removed declaration of RDC value) and reprinted from [7].

2.6 Neuro-MRI

Neurological MRI have been part of the background material in this thesis, and it contains challenges that arrises due to the small structures of the brain. In order to acquire high resolution images a smaller recieving coil known as a head coil is used, instead of the larger body coil seen in figure 2.7. This gives us a smaller FoV and thus increased resolution. A particular phantom from Philips that that is small enough to fit into this head coil is used to test for geometric distortions, SNR and uniformity deviation. For fMRI series one also tests for drift in the system, as well as changes in SNR over time, this is known as temporal SNR.

2.7 Geometric distortion requirements

Philips gives specifications on acceptable levels of geometric distortion in their MRI systems. These are dependent on field of view (FoV) used, and therefore dependent on the size of the phantom:

The phantom used in this thesis, the Philips body phantom, have the dimensions 40 cm diameter and falls just short of the defined sphere volume (DSV) at 53 cm in table 2.1. It is therefore expected that the recommended minimum deviation from linearity is slightly less than 1.4 % .

Table 2.1: Maximum accepted geometric distortion for two defined sphere volumes. Specified for Philips systems.

Defined Sphere Volume (DSV)	Gradient linearity
Inside a sphere with diameter 53 cm	<1.4 %
Inside a sphere with diameter 20 cm	<0.2 %

Chapter 3

Methods and Materials

3.1 Methodology

All programs in this thesis was written in Matlab R2012b (matrix laboratory, MathWorks).

In order to make a package with an assortment of analysis programs, several programs were implemented in a modular fashion and integrated through a common user interface. These programs would then later go on to be connected through a GUI masterprogram that gives the user options regarding which analysis would be performed and how information about the analysis should be stored.

In order to validate the accuracy of the automated software methods, and to what degree they differed from the manual analysis, image sets with known errors were utilized that where generated using a program based on the geometry of the phantom that was used at the time. This made it possible to verify the accuracy of the software for geometric distortion, and enabled statistical assessment of the expectations from manual and automatic analysis.

For the SNR and uniformity analysis new measurements were compared with results from existing QA-analysis tools utilizing a different program written in the program language IDL [19] currently used by the MR neuroimaging research group at OUS (Oslo University Hospital).

All test-results from the analysis package are written to a file in Excel format for further analysis and compatibility with existing QA methods. This format was chosen because it is easily accessible, and because it is allready used in

the manual analyses.

3.2 Materials

For all the tests analyzed and presented in this thesis two different phantoms where used. At first the geometric distortion head phantom called the ADNI phantom was utilized. Created by The Phantom Laboratory it is known as the Magphan Quantitative Imaging Phantom. It contains 165 polycarbonate spheres and was primarily chosen due to its ability to make assessments about 3 dimensional geometric distortions.

The ADNI phantom was later replaced with the much larger and 2 dimensional full body phantom from Philips that was available to us at Ullevål University Hospital, a 400 mm performance phantom. It is simpler in its construction as it contains no parts capable of breaking loose and is easier to place in the scanner in a stable manner due to a flat surface and the independency of different head coil design amongst vendors. It proved better for acquiring high contrast 2D images and made it possible to tailor the automated analysis to this specific phantom. This has become the phantom of choice at OUS for analysis of geometrical image distortion. It is also used for uniformity and SNR analysis, but it is not specifically better at these tasks than many other phantoms with uniform sections. For these types of recordings it is also possible to use plastic bottles filled with water up to 1.5 T, or mineral oil above 3 T.

Figure 3.1: The ADNI Magphan phantom.

Figure 3.2: The Philips body phantom.

3.3 Geometric Distortion method

The module for geometric distortion was designed to take input files in the DICOM format, more specifically DICOM images of phantoms that have a well known geometric structure. The Philips phantom is one of the phantoms that is usually used in the manual QA procedures, and because it has a uniform and known geometry, it was considered a good place to start testing automated methods.

The program reads .dcm files using the integrated functions dicomread and dicominfo, thus seperating the image from the header tags that accompany it. It has been strived towards utilizing the structure format that the Matlab environment provides for better handling of large sets of variables, and dicominfo provide this format as default.

Output variables of interest include but are not limited to:

- Geometric distortion values as per NEMA and IEC standards
- Object position deviation from expected x and y coordinates
- Image orientation
- Pixel spacing and Field of View
- Date, time, institution name and station name

Images from a previous QA instance were provided by my supervisors and other physicists at the university hospital. Figure 3.3 illustrates the execution scheme of the functions in the module.

After reading the images they are analyzed using the imfindcircles function. Though its use is reviewed here the method behind this function is explained in chapter 2.4.4. The imfindcircles function takes several input parameters, all of which must be individually adapted to the image that is being analyzed. This makes it necessary to have a basic understanding of the parameters in order to manually tweak them for use on a new MRI system. The parameters are named:

- Object polarity
- Edge threshold
- Sensitivity

Object polarity only takes the two values bright and dark. These indicate the difference in signal intensity between objects and background that the

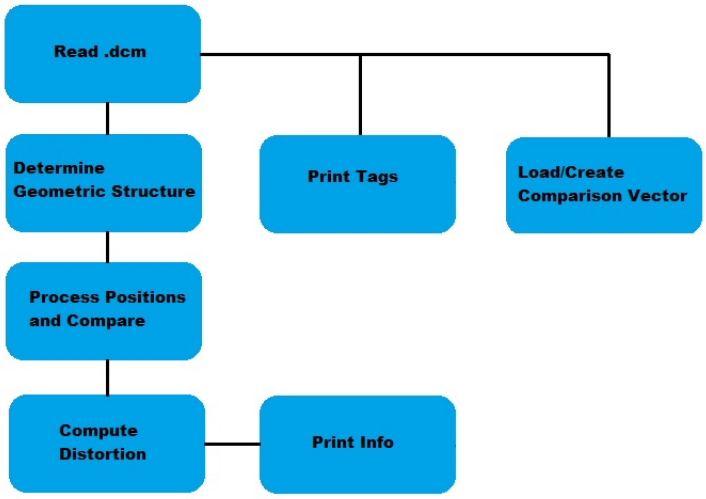


Figure 3.3: Execution scheme for the geometric distortion program module.

program will look for, bright objects have a higher intensity than the background while dark objects have lower intensity. Edge threshold sets the gradient threshold for edge pixels, that is the lowest change in intensity that will be considered as the edge of an object. The parameter takes values between 0 and 1, where values closest to 0 makes the program detect more objects with a softer edge. Finally the sensitivity parameter gives us the likelyhood of the internal voting system of the function, known as the accumulator array, giving us a positive outcome for detecting an object. All of these parameters must be correctly tuned in accordance with the image for there to be a high probability of detecting all objects and avoiding false positives. In addition there is a variable that takes a two element vector denoting the radii search area. This must be set to a suitable range, keeping in mind that larger ranges takes more computing time.

```
1 [centers_bright, radii_bright] = imfindcircles(img,[
    r_min r_max],'ObjectPolarity','bright', ...
2     'Sensitivity',Sens,'EdgeThreshold',Edge);
```

The output from the function are then two vectors, one listing positions of detected objects and another with the corresponding radii of each object. The program will then use a position given by the user that is an estimated center for the phantom. Using the detected object positions it then finds

the closest position to the user given position that is likely to be the actual phantom center. The method behind this is based of the geometry of the phantom, for example the Philips phantom has 9 objects in the center, set up like a square. As long as the user input is closer to the center object than any other object then these x,y-coordinates as found with imfindcircles will be defined as the center on which most length calculations are based.

The center coordinates have an inherent uncertainty due to the function imfindcircles. This is added to the uncertainty of the length between this and every other object, as found in the artificial image test, section 3.3.3.

3.3.1 False positive detection

False positives arise in many images analysed with many different settings. They can be seemingly circular objects such as the end points of the phantom fixtures, or high sensitivity combined with low search radii resulting in noise pixel clusters being registered as objects. Several algorithms have been written that test for the most common problems that arise in object detection in these kind of images. It is interesting to note that these algorithms take up most of the computing time in the geometric distortion module, and will remove detected objects in a live stream fashion for the user to observe. This is useful as an indicator as to what goes wrong in the case of objects being wrongfully removed.

In image 3.4 and 3.5 one sees that the two leftmost red circles, denoting registered objects, have been removed by the false positive detector. The blue crosses inside each circle in image 3.5 shows the center of the object as it is detected, and also shows that this object has passed all tests and is used in calculations. Note that the small object furthest down to the left has no such cross because it has not passed all tests. This is because this object is a part of the phantom fixture, and not a point of interest.

3.3.2 Validation against manual expert assessment

In order to test the module for geometric distortion and compare its values recieved from the program to that of manual methods, a blinded expert assessment was performed. Manual measurements by experienced MR phycisists is the gold standard for geometric distortion, this is therefore tested against the results from the QA program.

Figure 3.4: Geometric distortion image with each detetcted object marked by a red circle. Two false object detections have appeared around the leftmost part of the image were there is a phantom fixture with edges that have a circular appearance with a high image contrast.

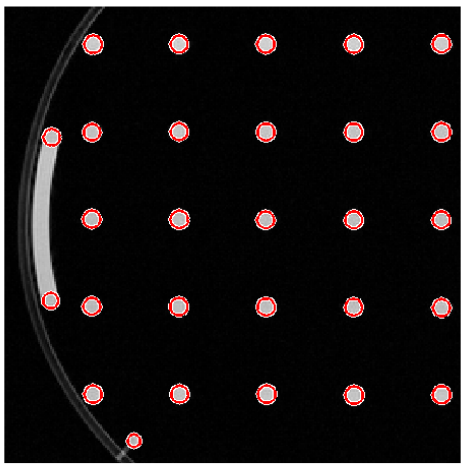
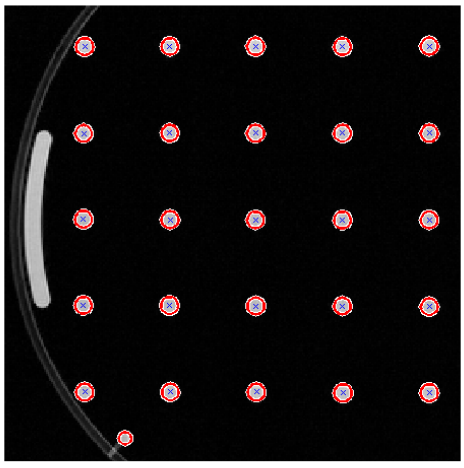


Figure 3.5: The same geometric distortion image as seen in figure 3.4. The leftmost false detections that were in the previous image have been removed by a false detection algorithm.



A set of artificially created images with a given distortion and noise level were established. 5 different physicists from the university hospital did then procede to do manual analysis of the images, according to the method described in section 3.6.2, without knowledge of what the expected results where.

Included among the images was a real MR-recording that had no known distortion level. The point of having a real image was to see the variance in the manual method in a real world scenario when performed by several different physicists.

The artificial images were created using the template for the Philips phantom, and then distorting it by adding a position dependent polynomial to the object matrix.

3.3.3 Synthetic distortion generation and analysis

The geometric distortion method was tested using a variation of the manual expert test. A program generated 100 images with known distortion values, the difference between these and program estimated values was calculated in order to create a boxplot, showing the distribution of estimation errors for the hundred images for each of the 45 objects found in the phantom.

The image distortion was set by a normalized random number generator that made a series of vectors. These were then used to decide the random distribution of distortions in each object throughout the synthetic phantom.

Noise in the images was made using the same normalized random number generator as before to create a noise matrix with peak signal of 5% of the image maximum signal.

```
% Create random noise. Rician distribution
noice_vec = 0.05*randn(dim(1),dim(2));
Image = Image + noice_vec;
```

3.4 Drift method

The module for drift analysis takes input files in the DICOM format, just like the geometric distortion module. However, since drift is a temporal

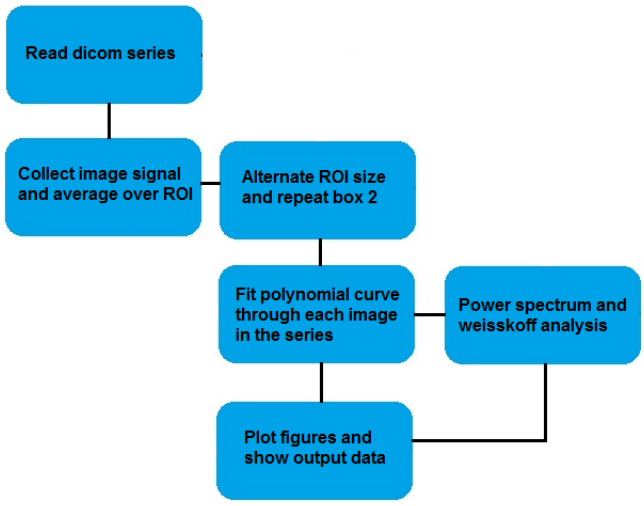


Figure 3.6: Execution scheme flowchart for the fMRI signal drift program module.

phenomenon that occurs in for example EPI recordings, one requires more than one image. Normally output from an EPI run results in hundreds, if not a couple of thousand images.

Module output consist of the following values:

- Drift
- Percent fluctuation
- Average signal
- Minimum signal
- Maximum signal
- temporal signal to noise ratio (tSNR)

For our testing of the method images of a homogenous bottle of mineral oil were taken, and images from older sessions were used as well.

Image 3.6 contains a flowchart illustrating the main events as they occur through the program.

For each pixel in the image a test was made in order to establish whether or

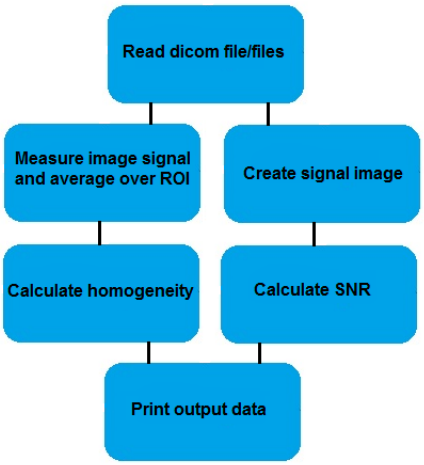


Figure 3.7: Execution scheme for the homogeneity program module.

not it was inside or outside of the ROI (region of interest), a circle defined by a radii. Pixel values outside of the ROI where discarded, while the ones inside where summed up and averaged.

For each image a set of varying radii was defined, default range of 1 to 20 pixels with steps of 1 pixel, and did ROI averaging for each of these radii. This makes this method the most time consuming by far. For the purposes of finding the drift value only one radii is necessary

Drift values and Weisskoff analysis was then calculated based on formula 2.20 and 2.21.

3.5 Uniformity method

Homogeneity was the final module in development and much like the drift analysis it takes pixel values in a given ROI for a certain DICOM image, but unlike when looking at drift only one single image is used here. The IEC developed standard of measuring homogeneity as seen in section 2.4.3 was used, equation 2.15. An option was added to include one additional image for spatial SNR analysis according to section 2.4.2.

The object detection uses Hough transformation through the imfindcircles function in Matlab, and finds an estimated center of gravity for the phantom in the image. Three different ROI are then placed on the phantom, with sizes either user defined or set to standard default values of 25, 50 and 85% of the total phantom area.

This module also contains the option to add an additional image and calculate spatial SNR values based on the equations in section 2.4.2. SNR and uniformity methods has many things in common, such as ROI positioning and pixel value scanning. It is for this reason that SNR and uniformity has been placed in the same module.

3.6 Current QA methods

The current QA used today at the Oslo University Hospital include analyses of the following parameters:

- Signal to noise ratio
- Uniformity
- Geometric distortion
- Point spread function

In this project the focus has been on the three first items in this list: SNR, uniformity and GD. In addition time series analysis for fMRI recordings has been added. The fMRI analysis does not have previous data for comparison with a manual analysis.

The current analysis for SNR and uniformity is automated using a program written by one of the supervisors of this Master thesis, Wibeke Nordhøy. Since there is currently no automated analysis for geometric distortion and because of an interest in switching to the programming language Matlab, the programs developed in this thesis contain overlapping functions. The programs have however been developed largely independently and comparisons of output have been made in the end phase of development to reveal sources of discrepancies. The IDL program is not described in detail in this thesis.

The rest of this section utilizes method descriptions found in the 2012 MRI QA analysis method book for OUS (BOQA).

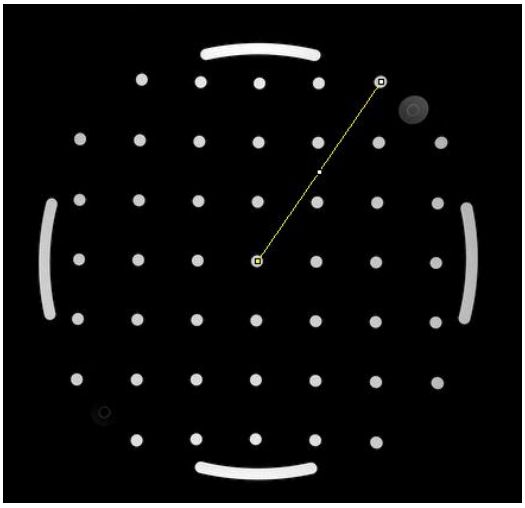


Figure 3.8: The manual analysis here shown using the utility program ImageJ. The distance between center and point 18 (NEMA numbering) is measured by placing two markers by hand and drawing a line between them.

3.6.1 SNR and uniformity

In the IDL program the object detection method differs from the Matlab version. In IDL the program finds a noise floor and the median of a line profile through the center of the image, for both x and y direction. The full width half maximum (FWHM) is then used to estimate a center of the object and the radius of the object to calculate the area and the different ROI sizes. SNR and uniformity is then found using the same methods as in section 2.4.2 and 2.4.3.

3.6.2 Manual analysis for geometric distortion determination

The current analysis used at the OUS for determining geometric distortion in phantoms is done manually using the programs ImageJ [20] and Microsoft Excel.

Each image consists of 45 circular objects, with 20 circles making up the outermost objects, see figure 2.8. The manual analysis includes only these 20 objects, while the new Matlab program utilizes all 45, which leads to a different numbering of the objects in the GD module between manual and automatic prosedures. When refering to the manual method of numbering

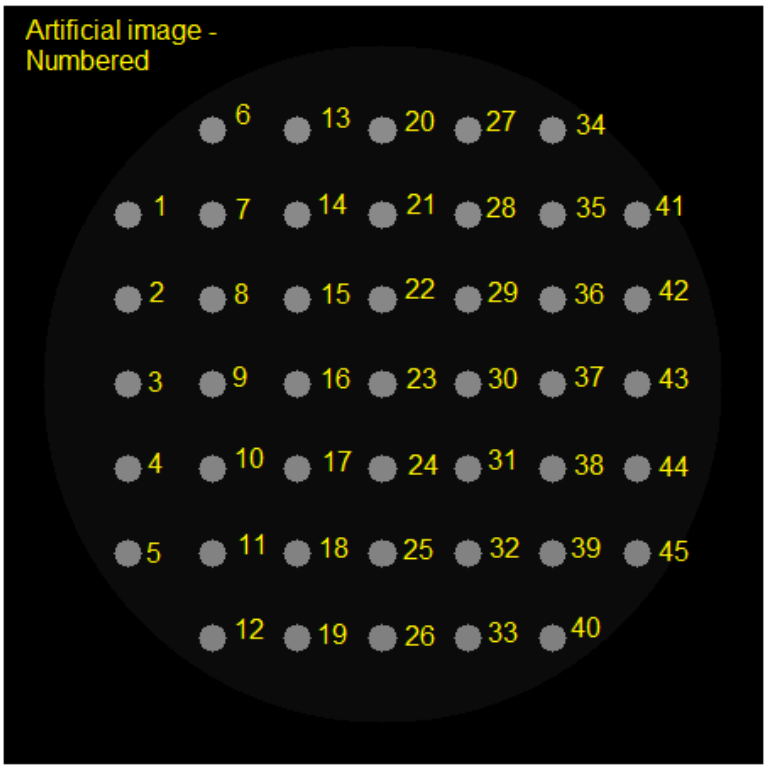


Figure 3.9: Software numbering of objects in the geometric distortion module.

there will be explicit use of the term 'NEMA numbering'.

Chapter 4

Results

This section contains several figures and tables with output from the QA program developed for this thesis.

4.1 Geometric distortion test

4.1.1 Manual geometric distortion test

Figure 4.1 shows results from the first synthetic image in the manual distortion test. The figure shows difference from known values of percentage distortion plotted against each object number from 1 to 20, following the NEMA enummeration method explained in section 3.6.2.

Table 4.1: Statistics of accuracy for Image1. Each expert is given a mean which represent the percent mean deviation from nominal object position, and a standard deviation (SD) and root mean square (rms) value for the percent deviation from nominal object positions for the entire data set.

Expert nr.	mean (%)	SD (%)	rms (%)
1	0.0347	0.1485	0.1488
2	-0.0968	0.1582	0.1821
3	0.1487	0.1389	0.2011
4	0.1396	0.2747	0.3019
5	0.1724	0.4508	0.472
Program	-0.0379	0.4414	0.4319

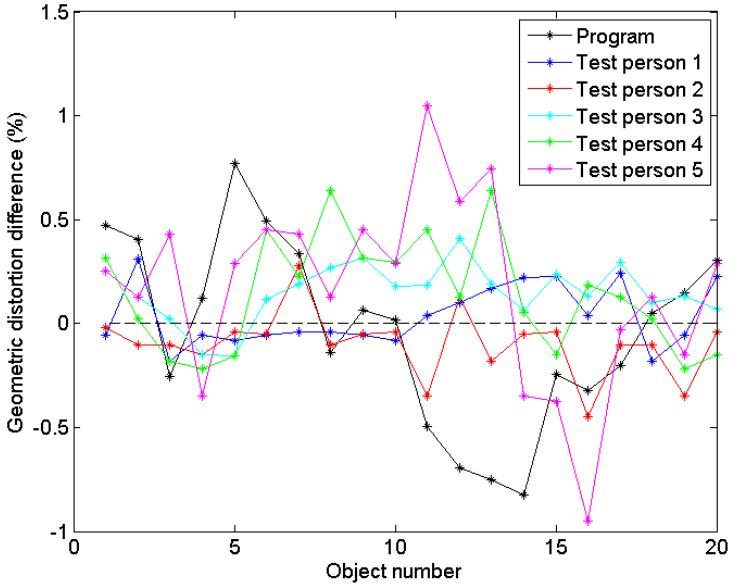


Figure 4.1: Error in object localization for each object number in image1 for each expert and the QA program. Objects are numbered according to NEMA.

An ANOVA table was made for image1 where the null hypothesis was that there was no difference between the data sets from the different experts and the data set from the QA program. Image 4.4 includes all experts and the program, and gives a p value of 0.0224. The null hypothesis is rejected at the 5 % level. This means that the different data sets are statistically different.

By removing one expert at a time and testing with new ANOVA tables, one may single out any data sets that cause the rejection of the null hypothesis. Removing expert 2 from the ensemble shows that there is no statistically significant difference between the remaining data sets. The new ANOVA table is seen in figure 4.5

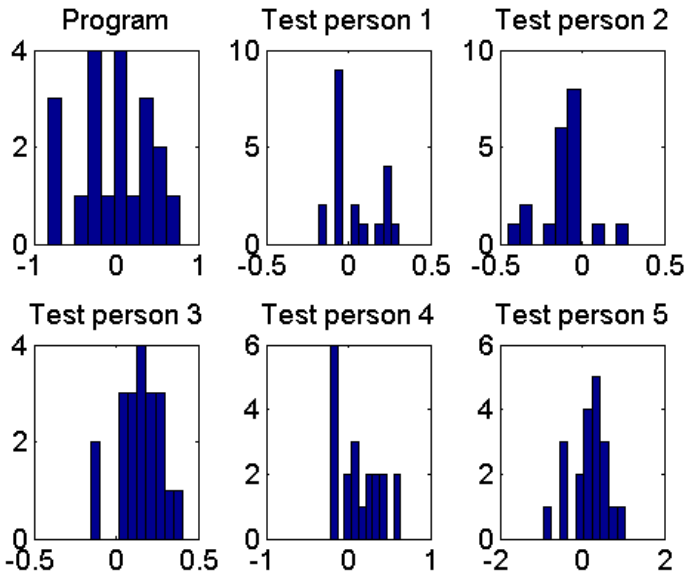


Figure 4.2: Histograms of measurements from the QA program and each expert for image1. The x axis is in measured geometric distortion percentage, y axis is the number of measurements.

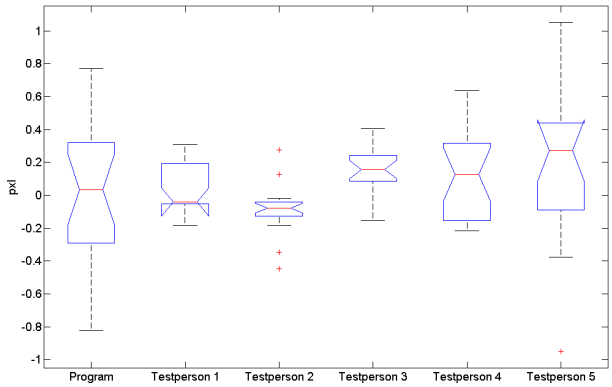


Figure 4.3: Boxplot representation of the values in figure 4.1.

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	1.2333	5	0.24666	2.74	0.0224
Error	10.2575	114	0.08998		
Total	11.4908	119			

Figure 4.4: Anova table using all experts as data sets for comparison from Image1. p value equals 0.0224, meaning that the null hypothesis is rejected at the 5 % level. The data sets are therefore statistically different.

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	0.6421	4	0.16053	1.56	0.1916
Error	9.7819	95	0.10297		
Total	10.424	99			

Figure 4.5: Anova table with expert 2 removed from the test ensemble of Image1. p value is equal to 0.1916, meaning that the null hypothesis is not rejected at the 5% level. The remaining data sets are therefore not statistically different.

Table 4.2: Statistics of accuracy for Image2. Each expert is given a mean which represent the percent mean deviation from nominal object position, and a standard deviation (SD) and root mean square (rms) value for the percent deviation from nominal object positions for the entire data set.

Expert nr.	mean (%)	SD (%)	rms (%)
1	0.0249	0.1864	0.1834
2	-0.0305	0.1885	0.1862
4	0.2030	0.3251	0.3763
5	0.0337	0.4444	0.4345
Program	-0.0031	0.4226	0.4120

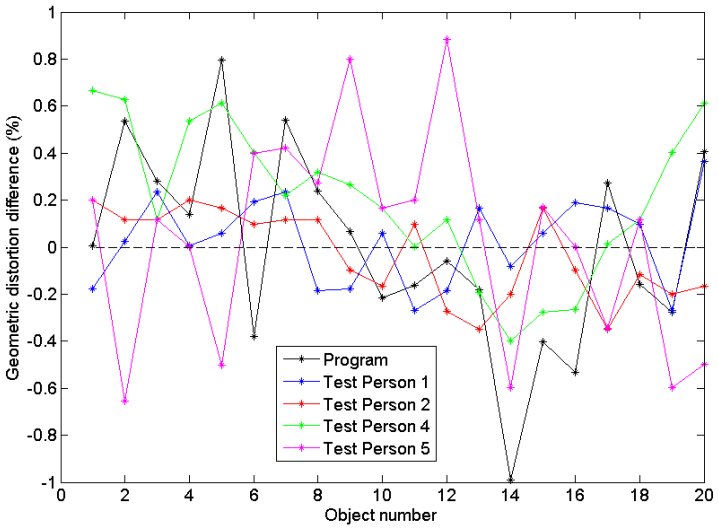


Figure 4.6: Error in object localization for each object number in image2 for each expert and the QA program. Objects are numbered according to NEMA.

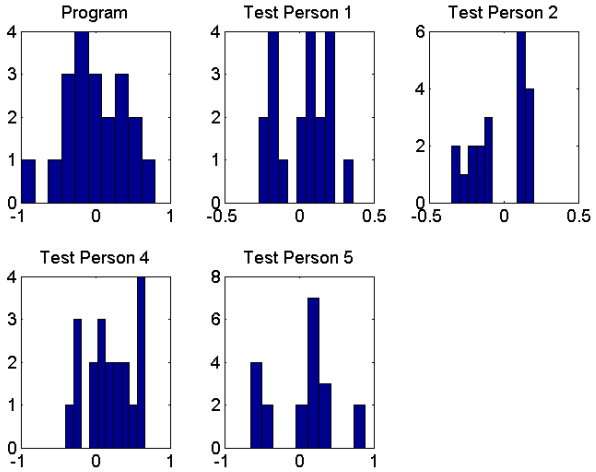


Figure 4.7: Histograms of measurements from the QA program and each expert for image2. The x axis shows measured geometric distortion percentage, y axis shows the number of measurements.

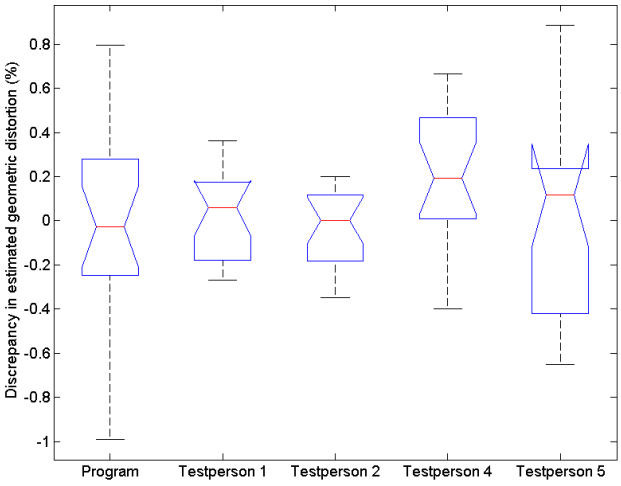


Figure 4.8: Boxplot representation of the values in figure 4.6.

An ANOVA table was also made for image2 with the same null hypothesis as for image1 (no difference between the different experts and the program). Image 4.9 includes all experts, and gives a p value of 0.2032. The null hypothesis is not rejected at the 5 % level.

Each participant was asked to log their time usage for each section of the analysis. The results can be seen in table 4.3. Using the Matlab program took approximately 1 minute per image, for both measurement and automatic analysis, for a total time of 3 minutes for the images shown here.

Table 4.3: Approximate time used by each participant in different sections of the manual analysis. The total time is the sum of measurement and analysis time.

Expert	Measure time (min)	Analysis time (min)	Total time (min)
1	12	20	32
2	16	8	24
3	12	20	32
4	8	20	28
5	16	12	28

ANOVA Table

Source	SS	df	MS	F	Prob>F
Groups	0.6704	4	0.16759	1.52	0.2032
Error	10.4895	95	0.11042		
Total	11.1598	99			

Figure 4.9: Anova table with all the data sets from Image2. p value is equal to 0.2032, meaning that the null hypothesis is not rejected at the 5% level. The data sets are therefore not statistically different

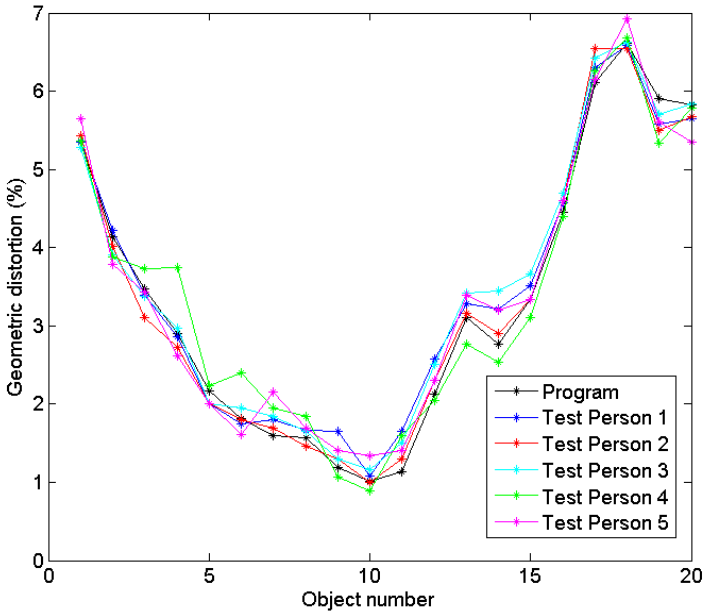


Figure 4.10: Geometric distortion values measured in a real MR image using the Philips full body phantom for both the program and the five experts.

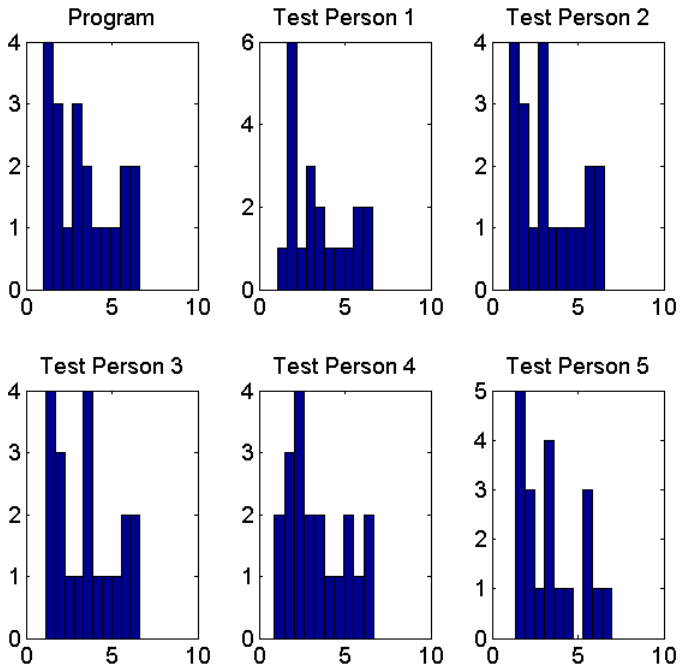


Figure 4.11: Histograms of measurements from the QA program and each expert for the real MR image using the Philips full body phantom. The x axis shows measured geometric distortion percentage, y axis shows the number of measurements.

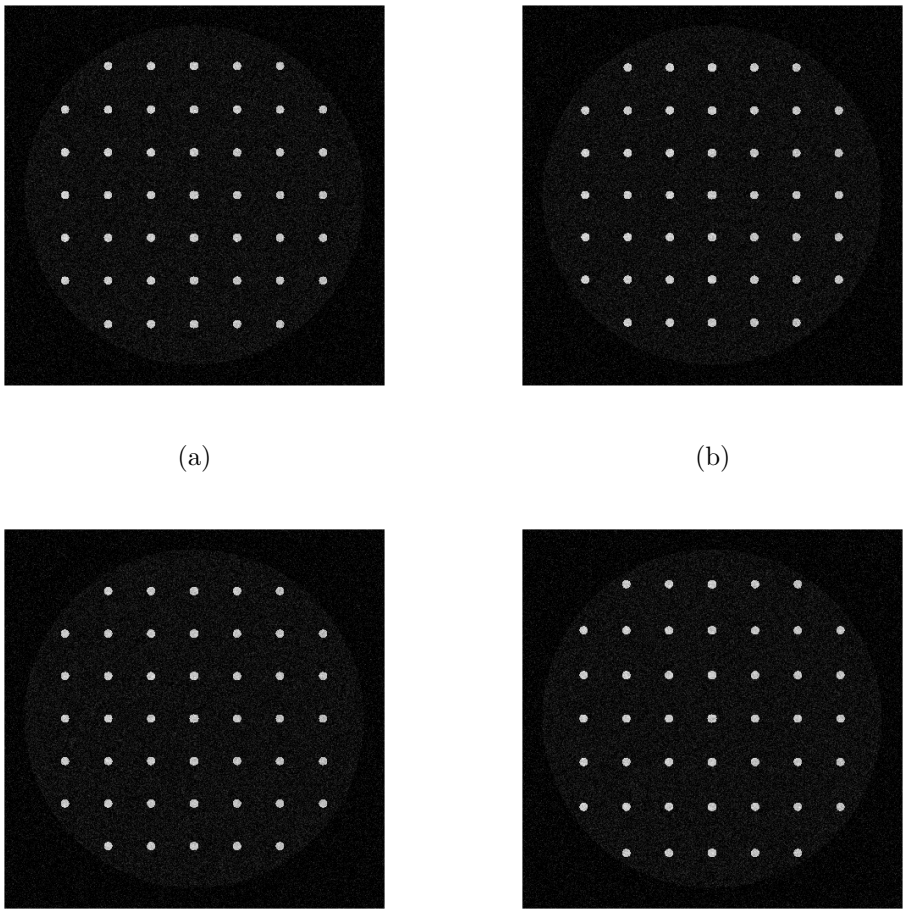
4.1.2 Synthetic geometric distortion test

The synthetic geometric distortion test used images created to resemble MR images, but with randomized noise and distortion values as explained in chapter 3.3.3. Figure 4.13a and figure 4.13b shows histograms for all 100 measurements for the first two objects in the phantom. The values are given as amount of pixel displacement from known object positions.

Figure 4.12 shows examples of the synthetic MR images. 4 randomly chosen images are shown along with distortion statistics represented in table 4.4.

Figure 4.13 shows histograms of the measurements made in the two first objects in the phantom, for 100 images. In figure 4.14 all object measurements are displayed as box plots. There are 100 measurements per object, which sums up to 4500 measurements. This figure uses pixel shift to denote the error in measured geometric distortion, while figure 4.15 uses error in measured geometric distortion percentage as it is reported during analysis.

A two-sided t-test on the mean values for each object in figure 4.14 was done. The null hypothesis is that the mean values of the objects are not statistically different from zero. In figure 4.16 the mean values are represented in a box plot. The null hypothesis is not rejected.



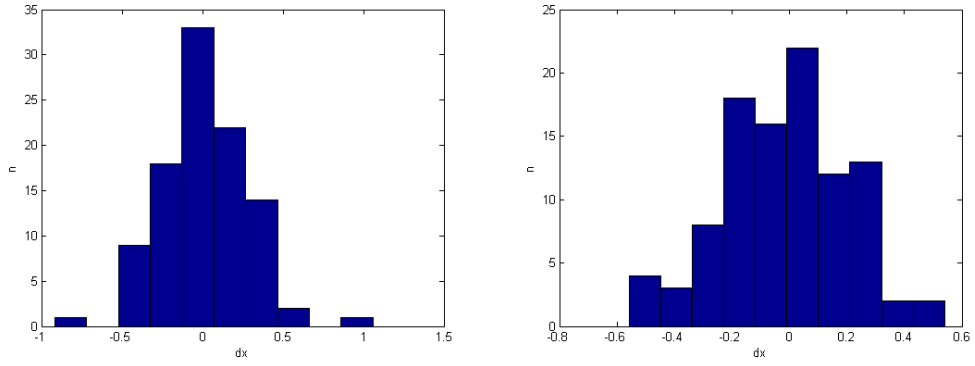
(a) (b)

(c) (d)

Figure 4.12: Four randomly selected images from the 100 synthetic images used for testing the geometric distortion module. The images contain small differences in GD that are relatively difficult to identify qualitatively.

Table 4.4: Mean, maximum, minimum, standard deviation (SD) and root mean square (rms) values measured by the program for four different images. Out of 100 images, four images were chosen as examples to show that the images in figure 4.12 are different though they look similar.

Image	Distortion denotation	mean	max	std	rms
a	dx (mm)	1.3485	2.5478	0.7374	1.5330
	dx (%)	0.9756	1.5242	0.4029	1.0538
b	dx (mm)	-0.5588	1.4320	0.4194	0.6959
	dx (%)	-0.5180	1.5988	0.4360	0.6740
c	dx (mm)	0.8661	2.4469	0.6738	1.0927
	dx (%)	0.6118	1.6313	0.4262	0.7429
d	dx (mm)	3.4579	8.9649	2.9111	4.4992
	dx (%)	2.5213	5.9766	1.7273	3.0454



(a) (b)

Figure 4.13: Examples of histograms of the 100 measurements for the two first measurement points in the phantom.

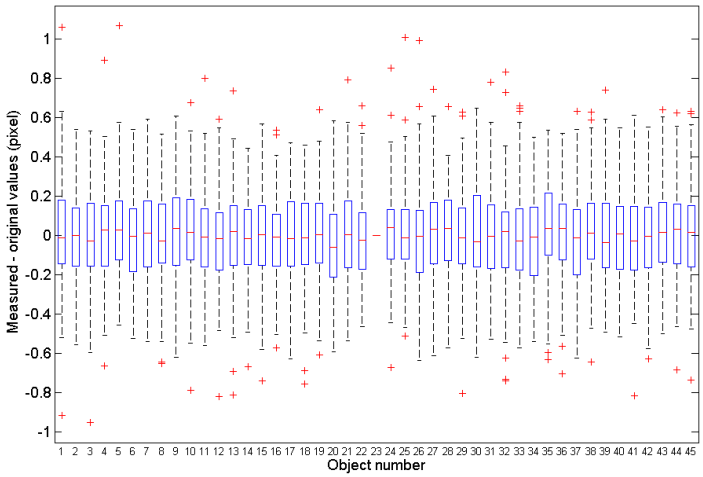


Figure 4.14: Box plot representation of distribution in difference between measured and actual values in pixel location for each of the measure points in the phantom. Each object number is measured in 100 different images.

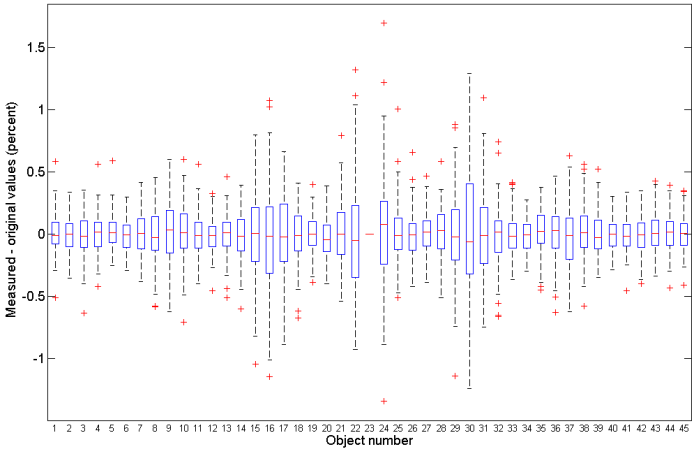


Figure 4.15: Box plot representation of distribution in difference between measured and known values in percent for each measure point in the phantom. Each object number is measured in 100 different images. Notice that accuracy in measurement is object number dependent when representing geometric distortion with percentages.

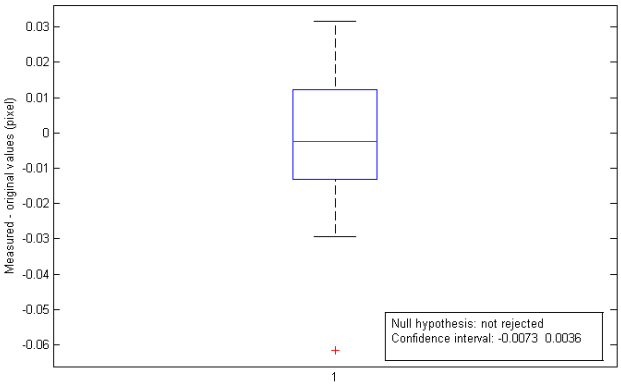


Figure 4.16: Box plot representation of mean values in figure 4.14. The null hypothesis is that the mean is no different from zero. The null hypothesis is not rejected at the 5 % level. The confidence interval is (-0.0073 0.0036).

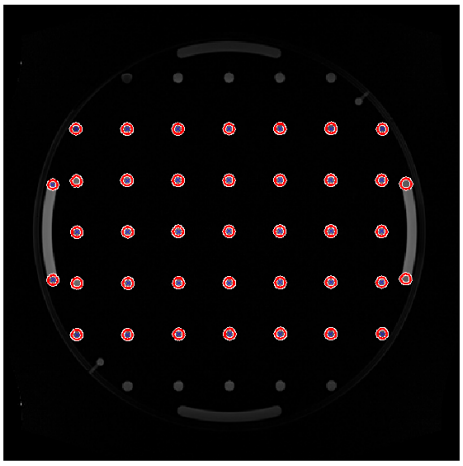
4.1.3 Geometric distortion image registration errors

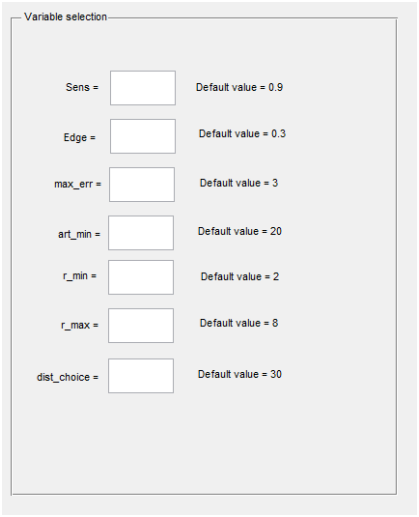
Figure 4.17 shows two different sets of optional settings that are used during scanning in the geometric distortion module, and the resulting object registrations. Low contrast images will require higher sensitivity and lower Edge values in order to register all measurement objects. Chosen values have to be set at a suitable level, since too high or low values will cause false positive or false negative detection of phantom markers.

4.1.4 3 dimensional geometric distortion visualization

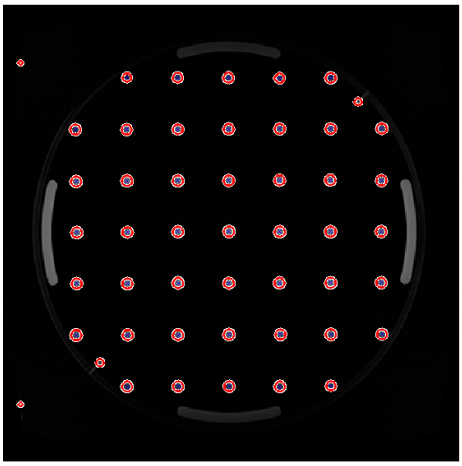
The GD module creates a mesh plot for visualizing the distribution of geometric distortion for each section in the phantom. As mentioned in section 2.3 every machine vendor have it's own mathematical correction algorithms that corrects for inhomogeneities in the static magnetic field. Four images are included in figure 4.18 that show examples of both images with and without geometric distortion correction algorithms, a and c, b and d respectively. The x and y axis in the image are number of measurement points away from the center of the phantom.

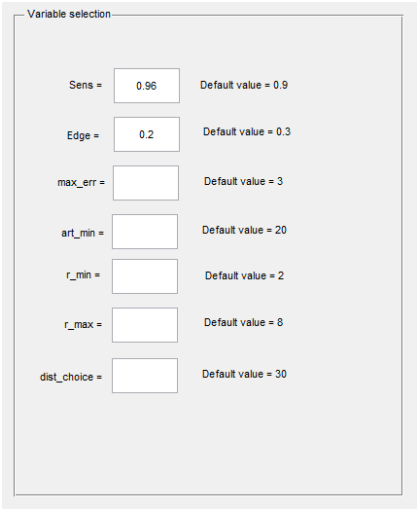
Figure 4.19 shows two mesh images of the distortion in the phantom used in the manual GD tests in section 4.1.1. Note that the artificial images in 4.19 have small random variations in geometric distortion due to the method of image generation, though it is of similar value in most directions. The real MR phantom images in figure 4.18 are randomly distorted.





(a) (b)





(c) (d)

Figure 4.17: Examples of one image scanned with two different option settings. Figure a shows default settings. Due to low contrast in the image periphery some objects are undetected. Figure c shows the same image, but scanned with a higher sensitivity and a lower edge treshold. This causes more objects to be detected, but also causes false detections in the leftmost parts of the image.

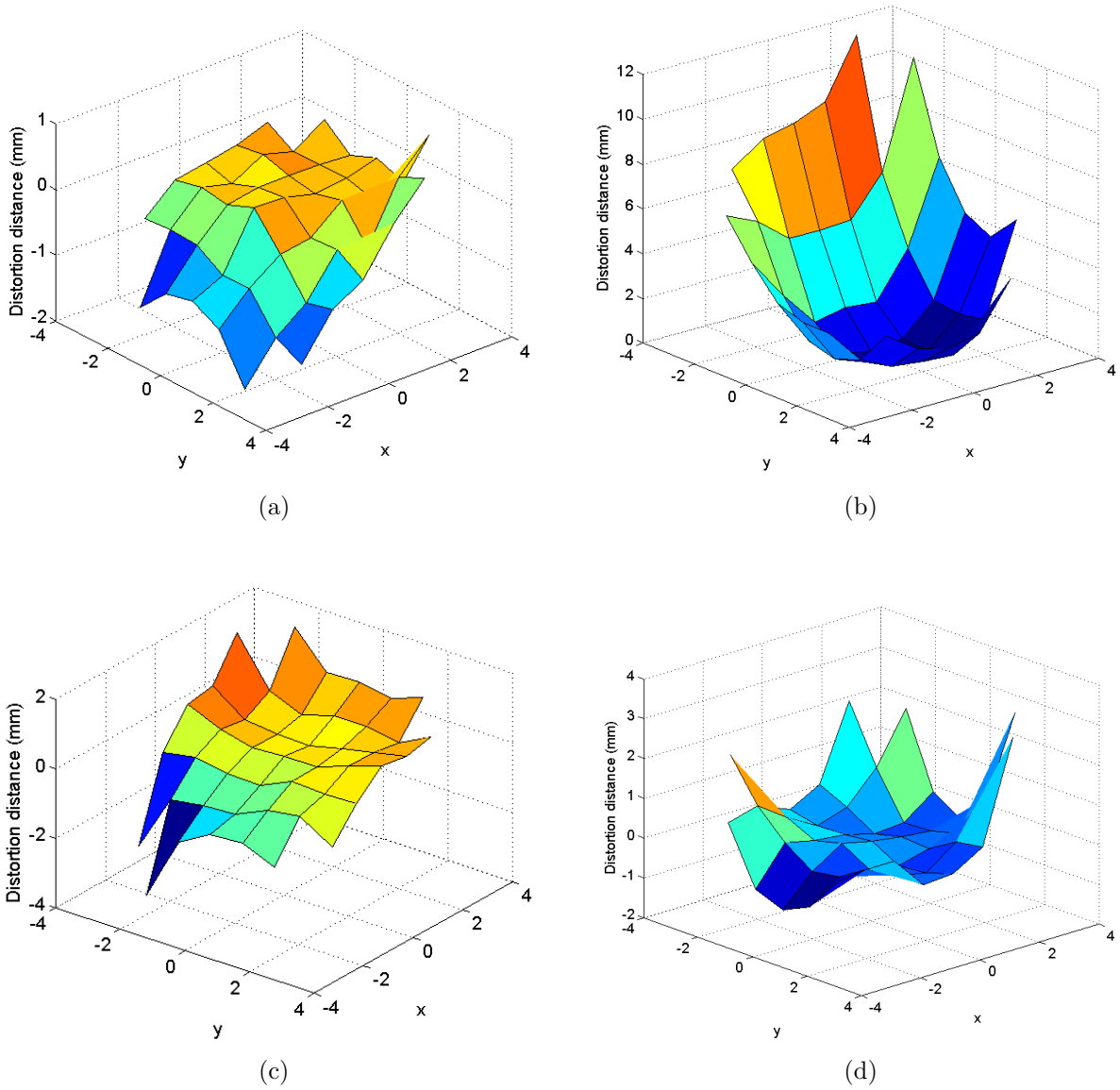


Figure 4.18: Example images of mesh plots of geometric distortion in real MR images. Images a and c used correction algorithms, while image b and d did not. x and y axis are the number of measurement points away from the center of the phantom.

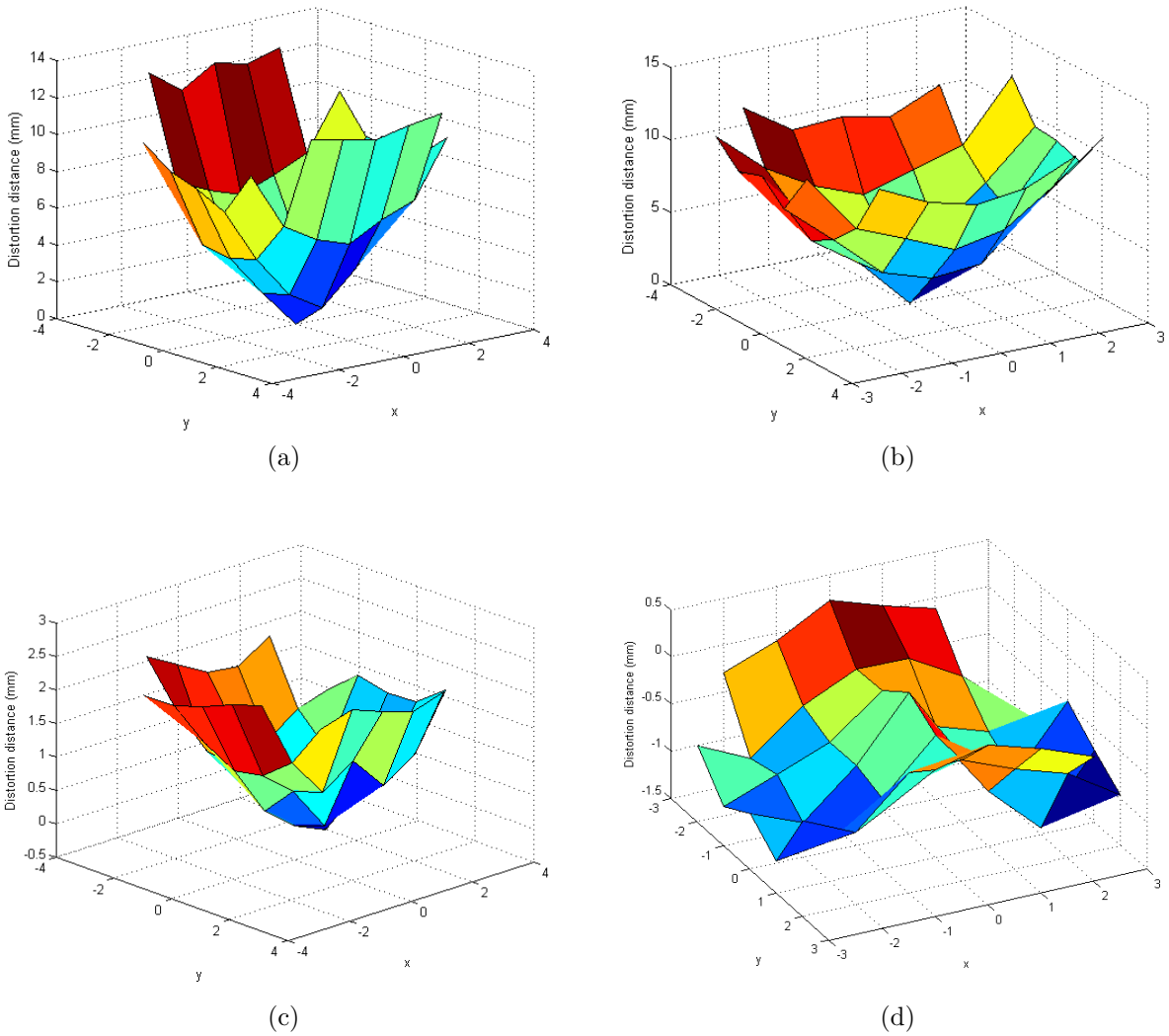


Figure 4.19: Mesh images of GD in Image1 (a) and Image2 (b) used in the manual GD test in section 4.1.1 and image a (c) and Image b (d) from figure 4.12. x and y axis are the number of measurement points away from the center of the phantom.

4.2 Image stability

Figure 4.20 shows an example of mean signal temporal drift in an fMRI series of 100 images, with an ROI radius of 10 pixels. The drift value is 1.6422 and percent fluctuation is at 0.29922. The green line is a second degree polynomial fit to the data.

In figure 4.21 there is an example of measured and theoretical CV values, see equation 2.21, in a so called Weisskoff plot. The measured and theoretical values should follow for as long as possible, but seem to depart relatively early.

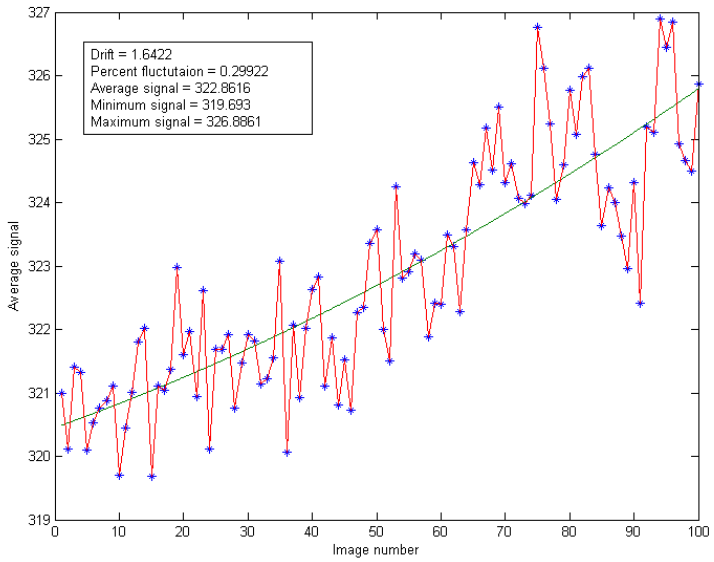


Figure 4.20: ROI mean signal plotted for each image in a fMRI series. ROI radius were 10 pixels.

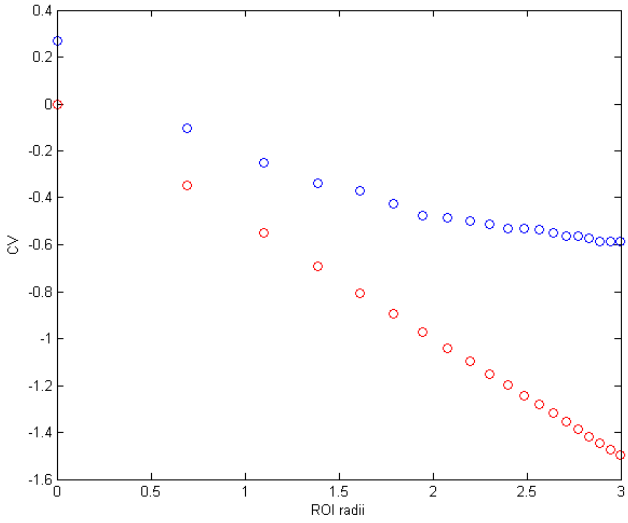


Figure 4.21: Sample Weisskoff plot. In this figure the blue curved line of dots is the measured CV, while the red straight line of dots is the theoretical CV. The x and y axis are both logarithmic.

4.3 Analysis comparisons on existing QA data

4.3.1 SNR

Table 4.5 shows output values for SNR1 and SNR2, both for the Matlab program (QAP) developed for this thesis and for the IDL program currently in use for QA assessments by the MR physics group of the Intervention Centre at the OUS. The table is divided into two main sections, one for each brand, Aleris Siemens Avanto and GE-450. The analysis year shows which program was used, as all measurements taken in 2012 used IDL, while Matlab was used for all data anlyses in 2014. Both the new and old QA sessions used the same images and percentage of area for their ROI. For GE-450 the original images for the transversal direction that were taken in 2012 were missing.

4.3.2 Uniformity

Several datasets from 2012 with previously analysed images were re-analysed using the Matlab program. The comparisons can be seen in table 4.6. There are three different MRI scanners: Siemens Avanto, GE-450 and GE Signa HDxt (3T). The uniformity U has been found according to equation 2.15 (QAP) and using the IDL program explained in section 2.4.3. The right most value in the table, Diff, is the difference U(QAP) - U(IDL).

4.3.3 Geometric distortion

Table 4.7 shows comparisons between manual and QAP measurements for various MR scanner vendors and several scan directions.

Table 4.5: Table showing comparisons between SNR measurements taken both with the IDL program (Old method), and the QAP. Two different MRI brands, three image directions, Transversal, Sagittal and Coronal. Three ROI sizes, 25, 50 and 85 % . And two SNR types, SNR1 and SNR2.

Brand	Year	SNR type	ROI area (%)	Trans	Sag	Cor
Siemens Avanto	2012 (Old)	SNR1	25	243	281	284
	2014 (QAP)			242	284.6	291.9
	Diff (%)			-0.41	1.28	2.78
	2012 (Old)		50	249	273	278
	2014 (QAP)			243.5	281.9	284.3
	Diff (%)			-2.21	3.2	2.27
	2012 (Old)		85	256	253	257
	2014 (QAP)			255	268.6	266.9
	Diff (%)			-0.39	6.16	3.85
	2012 (Old)	SNR2	25	258	302	307
	2014 (QAP)			239.9	283.9	287.2
	Diff (%)			-7.02	-5.99	-6.45
	2012 (Old)		50	267	306	313
	2014 (QAP)			252.7	296	298
	Diff (%)			-5.36	-3.27	-4.79
	2012 (Old)		85	282	300	308
	2014 (QAP)			273.8	300	307.2
	Diff (%)			-2.91	0	-0.26
GE-450	2012 (Old)	SNR1	25	131	177	164
	2014 (QAP)			-	175.9	178.2
	Diff (%)			-	-0.62	8.66
	2012 (Old)		50	130	175	150
	2014 (QAP)			-	177.7	164.9
	Diff (%)			-	1.54	9.93
	2012 (Old)		85	129	160	138
	2014 (QAP)			-	168.2	143.3
	Diff (%)			-	5.13	3.84
	2012 (Old)	SNR2	25	155	213	215
	2014 (QAP)			-	191.9	198.8
	Diff (%)			-	-9.9	-7.53
	2012 (Old)		50	155	214	208
	2014 (QAP)			-	209	210.6
	Diff (%)			-	-2.37	1.25
	2012 (Old)		85	156	198	187
	2014 (QAP)			-	204.3	193.6
	Diff (%)			-	3.18	3.53

Table 4.6: Table showing uniformity measurements using QAP and IDL, and the difference between them. Three different brands: Siemens Avanto, GE-450 and GE Signa HDxt (3T). Three different image directions: Coronal, Sagittal and Transversal. Also three different ROI sizes: 25, 50 and 85 % .

Brand	Image	ROI (%)	U (QAP)	U (IDL)	Diff
Siemens Avanto (1.5 T)	Cor	25	97.85	98	-0.25
		50	93.68	94	-0.32
		85	82.65	83	-0.35
	Sag	25	97.29	97	0.29
		50	93	93	0
		85	83.35	84	-0.65
	Tra	25	98.96	99	-0.04
		50	97.5	98	-0.5
		85	94.5	95	-0.5
GE-450 (1.5 T)	Cor	85	85.87	86.1	-0.23
	Sag	85	88.27	88.6	-0.33
GE Signa HDxt (3T)	Cor	25	97.71	97.6	0.11
		50	96.47	95.9	0.57
		85	94.63	94.1	0.53
	Sag	25	97.1	96.9	0.2
		50	95.55	95	0.55
		85	93.1	92.8	0.3
	Tra	25	98	98	0
		50	97.69	97.6	0.09
		85	97.32	97	0.32

Table 4.7: Table showing the difference between GD measurements using the manual analysis and the QAP. The columns named manual and Matlab are showing the maximum values of GD in an image. The column Image shows image direction, and the column MR system shows the MR scanner vendor.

MR system	Image	Manual GD (%)	QAP GD (%)	Difference (%)
Siemens Avanto (1.5 T)	Cor	1.14	1.135	0.005
	Sag	1.11	0.84	0.27
	Tra	1.22	1.55	-0.33
GE Signa HDxt (3T)	Cor	1.4	1.17	0.23
	Tra	3.1	3.1	0

4.4 Graphic User Interface

Figure 4.22 shows the opening menu of the program, with five alternatives. SNR and Uniformity which opens the homogeneity module in figure 4.23, Dynamic Stability which opens the time series drift module shown in figure 4.25 and Geometric distortion which is shown in figure 4.24. The two last options are documentation and exit. Documentation shows comments about the program, such as name and contact information of the author, time of writing and Matlab version number used.

Each GUI window have several variables open for editing before analysis, a viewing window for DICOM files, file harddrive location notifier and a convenient back button for selecting other modules. They also have a varying number of output variables that are visisble in the GUI, while simultaneously printing to an MS Excel file chosen by the operator.

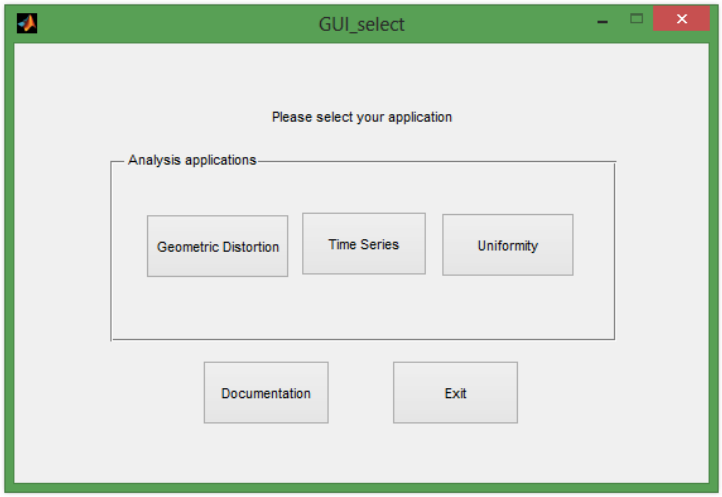


Figure 4.22: Application selection graphical user interface for Matlab based QA program. Three available applications for QA methods, two buttons for the auxillary options documentation and exit.

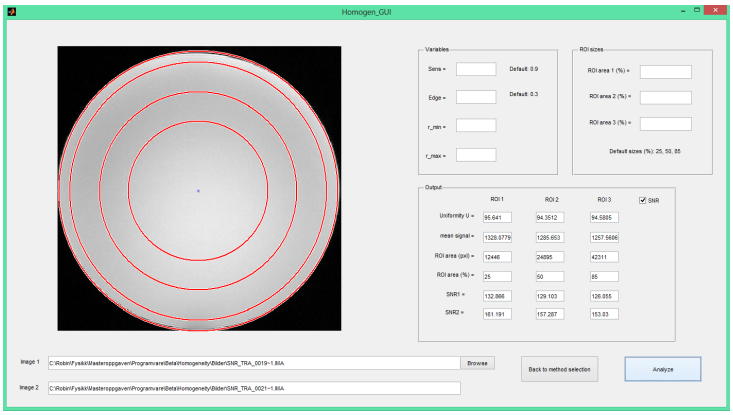


Figure 4.23: Uniformity and SNR graphical user interface. Left hand side of the window shows the image loaded from the current DICOM file, which is selected using the browse function in the lower part of the window. Right hand side allows for selecting different variable values and ROI sizes. The bottom right part gives analysis output for quick access, which are additionally stored in an MS Excel file for further use.

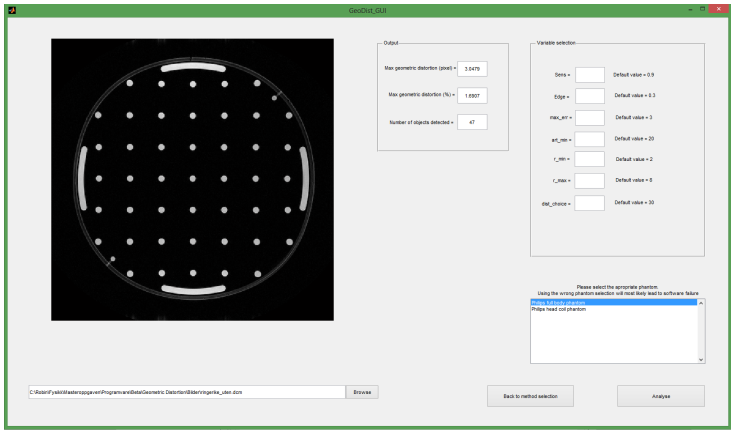


Figure 4.24: Geometric distortion graphical user interface. Left hand side of the window shows the image loaded from the current DICOM file, which is selected using the browse function in the lower part of the window. Right hand side allows for selecting different variable values and phantom structures.

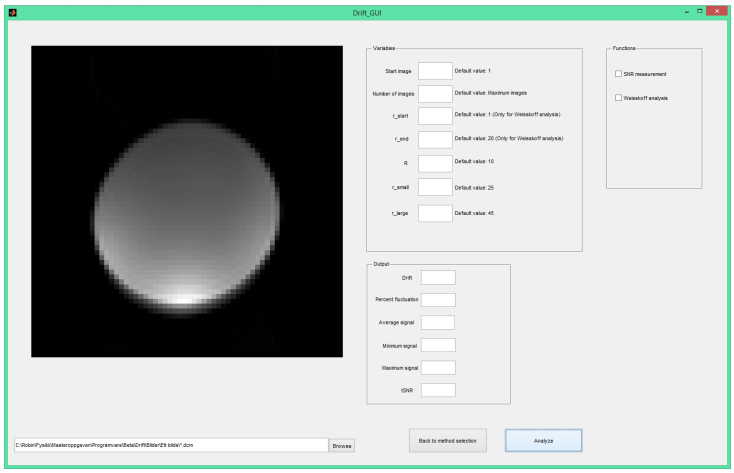
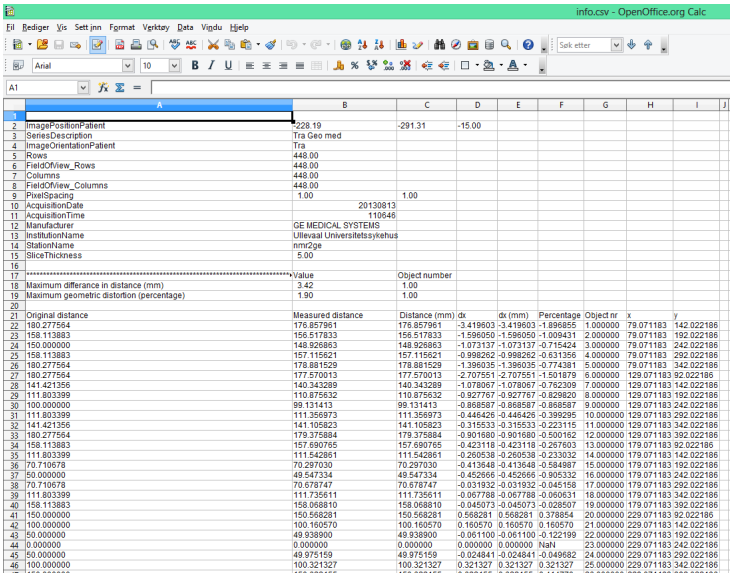
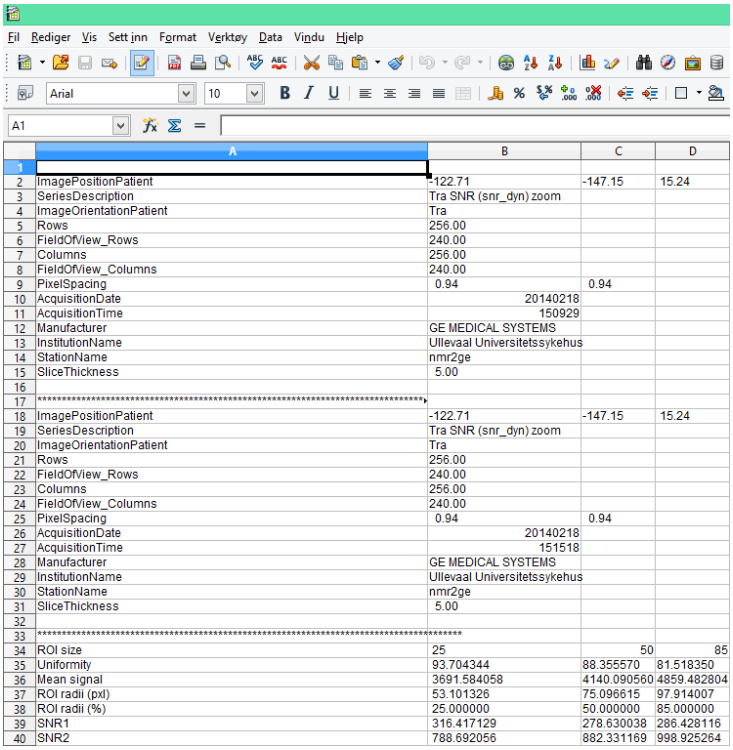


Figure 4.25: Time series graphical user interface. Left hand side of the window shows an image loaded from the current folder, which is selected using the browse function in the lower part of the window. Right hand side allows for selecting different variable values and functions such as Weisskoff analysis and temporal SNR.



	A	B	C	D	E	F	G	H	I	J
1										
2	ImagePositionPatient	-228.19	-291.31	-15.00						
3	SeriesDescription	Tra Geo med								
4	ImageOrientationPatient	Tra								
5	Rows	448.00								
6	FieldOfView_Rows	448.00								
7	Columns	448.00								
8	FieldOfView_Columns	448.00								
9	PixelSpacing	1.00	1.00							
10	AcquisitionDate	20130813								
11	AcquisitionTime	110646								
12	Manufacturer	GE MEDICAL SYSTEMS								
13	InstitutionName	Ullevaal Universitetssykehus								
14	StationName	nmr2ge								
15	SliceThickness	5.00								
16										
17	**********	Value	Object number							
18	Maximum difference in distance (mm)	3.42	1.00							
19	Maximum geometric distortion (percentage)	1.90	1.00							
20										
21	Original distance	Measured distance	Distance (mm)	dx	dx (mm)	Percentage	Object nr	x	y	
22	180.277564	176.857961	176.857961	-3.419603	-3.419603	-1.896855	1.000000	79.071183	142.022186	
23	158.113883	156.517833	156.517833	-1.596050	-1.596050	-1.009431	2.000000	79.071183	192.022186	
24	150.000000	148.926863	148.926863	-1.073137	-1.073137	-0.715424	3.000000	79.071183	242.022186	
25	158.113883	157.115621	157.115621	-0.998262	-0.998262	-0.631356	4.000000	79.071183	292.022186	
26	180.277564	178.881529	178.881529	-1.396035	-1.396035	-0.774381	5.000000	79.071183	342.022186	
27	180.277564	177.570013	177.570013	-2.707551	-2.707551	-1.501879	6.000000	129.071183	92.022186	
28	141.421356	140.343289	140.343289	-1.078067	-1.078067	-0.762309	7.000000	129.071183	142.022186	
29	111.803399	110.875632	110.875632	-0.927767	-0.927767	-0.829820	8.000000	129.071183	192.022186	
30	100.000000	99.131413	99.131413	-0.868587	-0.868587	-0.868587	9.000000	129.071183	242.022186	
31	111.803399	111.356973	111.356973	-0.446426	-0.446426	-0.399295	10.000000	129.071183	292.022186	
32	141.421356	141.105823	141.105823	-0.315533	-0.315533	-0.223115	11.000000	129.071183	342.022186	
33	180.277564	179.375884	179.375884	-0.901680	-0.901680	-0.500162	12.000000	129.071183	392.022186	
34	158.113883	157.690765	157.690765	-0.423118	-0.423118	-0.267603	13.000000	179.071183	92.022186	
35	111.803399	111.542861	111.542861	-0.260538	-0.260538	-0.233032	14.000000	179.071183	142.022186	
36	70.710678	70.297030	70.297030	-0.413648	-0.413648	-0.584987	15.000000	179.071183	192.022186	
37	50.000000	49.547334	49.547334	-0.452666	-0.452666	-0.905332	16.000000	179.071183	242.022186	
38	70.710678	70.678747	70.678747	-0.031932	-0.031932	-0.045158	17.000000	179.071183	292.022186	
39	111.803399	111.735611	111.735611	-0.067788	-0.067788	-0.060631	18.000000	179.071183	342.022186	
40	158.113883	158.068810	158.068810	-0.045073	-0.045073	-0.028507	19.000000	179.071183	392.022186	
41	150.000000	150.568281	150.568281	0.568281	0.568281	0.378854	20.000000	229.071183	92.022186	
42	100.000000	100.160570	100.160570	0.160570	0.160570	0.160570	21.000000	229.071183	142.022186	
43	50.000000	49.938900	49.938900	-0.061100	-0.061100	-0.122199	22.000000	229.071183	192.022186	
44	0.000000	0.000000	0.000000	0.000000	0.000000	NaN	23.000000	229.071183	242.022186	
45	50.000000	49.975159	49.975159	-0.024841	-0.024841	-0.049682	24.000000	229.071183	292.022186	
46	100.000000	100.321327	100.321327	0.321327	0.321327	0.321327	25.000000	229.071183	342.022186	

Figure 4.26: Example image of output file written from the geometric distortion module. Only a part of the document is visible in this figure.



	A	B	C	D
1				
2	ImagePositionPatient	-122.71	-147.15	15.24
3	SeriesDescription	Tra SNR (snr_dyn) zoom		
4	ImageOrientationPatient	Tra		
5	Rows	256.00		
6	FieldOfView_Rows	240.00		
7	Columns	256.00		
8	FieldOfView_Columns	240.00		
9	PixelSpacing	0.94	0.94	
10	AcquisitionDate	20140218		
11	AcquisitionTime	150929		
12	Manufacturer	GE MEDICAL SYSTEMS		
13	InstitutionName	Ullevaal Universitetssykehus		
14	StationName	nmr2ge		
15	SliceThickness	5.00		
16				
17	**********			
18	ImagePositionPatient	-122.71	-147.15	15.24
19	SeriesDescription	Tra SNR (snr_dyn) zoom		
20	ImageOrientationPatient	Tra		
21	Rows	256.00		
22	FieldOfView_Rows	240.00		
23	Columns	256.00		
24	FieldOfView_Columns	240.00		
25	PixelSpacing	0.94	0.94	
26	AcquisitionDate	20140218		
27	AcquisitionTime	151518		
28	Manufacturer	GE MEDICAL SYSTEMS		
29	InstitutionName	Ullevaal Universitetssykehus		
30	StationName	nmr2ge		
31	SliceThickness	5.00		
32				
33	**********			
34	ROI size	25	50	85
35	Uniformity	93.704344	88.355570	81.518350
36	Mean signal	3691.584058	4140.090560	4859.482804
37	ROI radii (pxl)	53.101326	75.096615	97.914007
38	ROI radii (%)	25.000000	50.000000	85.000000
39	SNR1	316.417129	278.630038	286.428116
40	SNR2	788.692056	882.331169	998.925264

Figure 4.27: Example image of output file written from the uniformity and SNR module. This figure shows the entire document.

Chapter 5

Discussion

5.1 Program development and resulting structure

The program that has been developed is shown in its completed state in the appendix. With the exception of the GUI code.

The main philosophy behind the development procedure was to cover several important QA methods used today in MRI, and lay a foundation that others could add to or improve upon. To ensure that one does not have to know everything about this program package in order to add a function all the main modules are structurally independent and seperated from each other and use locally defined variables that do not cross over from one module to the other.

As seen in section 4.4 the program contains a graphical user interface (GUI) that was developed using feedback from both my supervisors and other phycisists at the OUS. This is intended to make the software more user friendly, and make variable options and underlying methods more obvious. It is still assumed that some level of training will be neccessary in order to use the software effectively. Figure 5.1 shows the final program structure hierarchy with the top two levels of the structure containing only GUI programs.

So far the analysis performed by the program encompasses uniformity, spatial and temporal SNR, and geometric distortion. The latter is currently

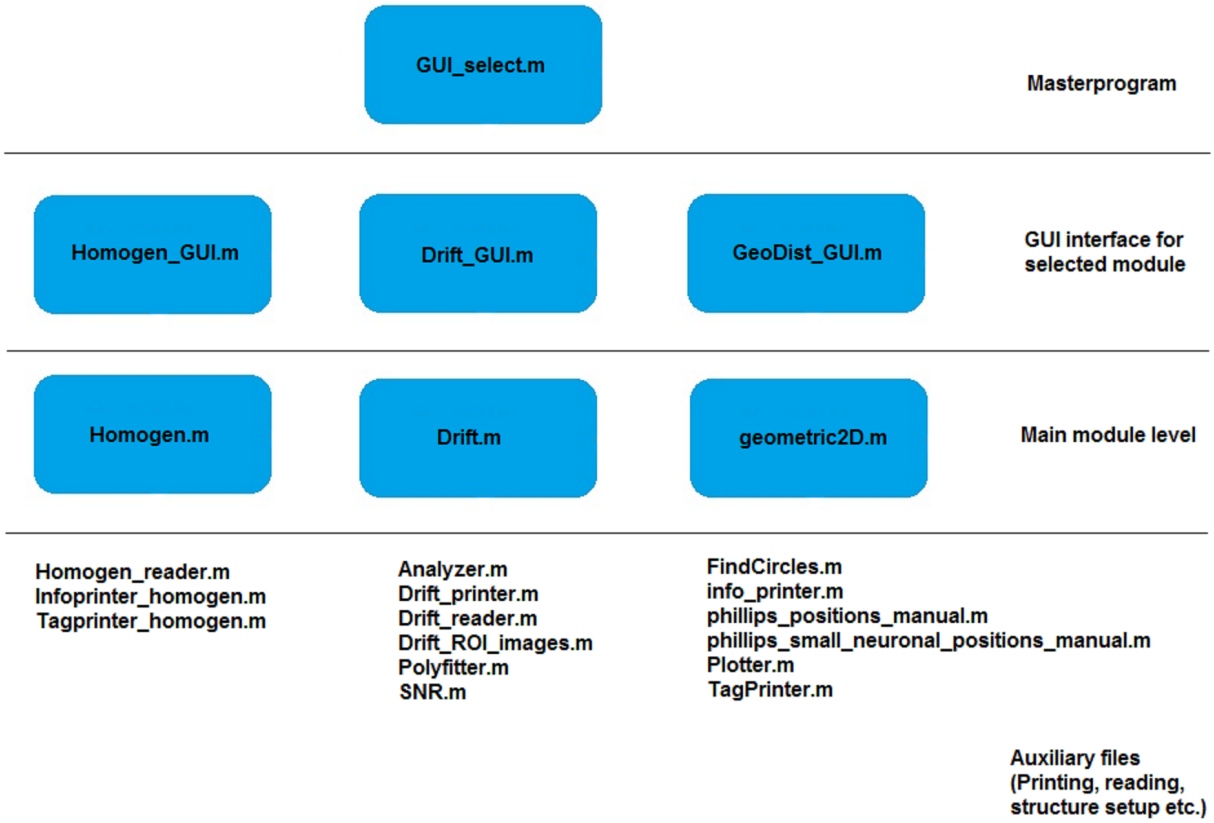


Figure 5.1: Program package structure. A level oriented structure image naming each program in the thesis. All interaction with the software goes through the program GUI_ select.m, which in turn activates the appropriate graphical user interface.

restricted to two phantoms, one large body phantom and one smaller used with a head coil, both produced by Philips Healthcare. Adding new phantoms in the future should be relatively simple for one who is proficient in Matlab, but care must be taken to follow and understand the GD module method, especially if the new phantom has a drastically different geometry. In the beginning of this thesis it was thought that the ADNI phantom, as discussed in section 3.2, could be used instead of the one from Philips, due to the possibility for 3-dimensional geometric distortion measurements. This idea was halted due to problems with the phantom. The images taken with the phantom were of low contrast and early tests with QAP proved sporadically unsuccessful at locating the measurement points in the phantom. If this phantom is added to the current program it will require that several false

object detection procedures are turned off for a successful reading due to the way the program orients itself geometrically. In order to successfully integrate a new phantom structure it will be neccessary to write a new m-file that addresses the variables found in the program "philips_positions_manual.m" and initialize it using the GUI.

5.2 Dataset comparisons for overall program modules

5.2.1 Geometric distortions

In clinical data one rarely has the definitive answer to how much geometric distortion is present in an image: Even when using a phantom with known dimensions the measured distortion is still prone to both random and systematic error. In order to compare the program with manual analysis there was a need for images with known geometric distortions. This was solved by generating synthetic images, section 3.3.2, with a known GD and then compare the performance of the program as well as experts (with long experience in measuring GD) with the ground truth. This is illustrated in figure 4.1 and 4.6.

For geometric distortion our goal was to show that a program could perform at least as well as a human analysis, and complete the task much faster and with less manual labour than with current methods. During the manual method test, section 3.6.2 and 4.1.1, the participants where asked to find the length of each of 20 objects (NEMA numbering) from phantom center in each image, calculate maximum GD and log their time usage. Tables 4.1 and 4.2 shows that the root mean square (rms) of most manual measurements where better than the automatic measurements, with the exception of expert 5 who did worse than the program in both images. Figure 4.7 shows comparisons between measurements from 2012 and new ones using the program in this thesis. The discrepancies are small. The number of experts and image test cases used in this trial is not large enough to conclude that there is a real difference between manual and automatic methods, but there is an indication that none of them perform worse on average, and the program performed at least as well as the experts on average.

Figure 4.10 shows the geometric distortion values collected for the real MR image using both manual and automatic procedures. No definitive GD values

exist for this image. None of the procedures seem to stand out as obviously different from the others, and the program follows a curve similar to the mean of the collective curves.

In order to expand available measurement data and make statistical inference about the accuracy of the GD module the same synthetic image generator as in the manual methods test was used and expanded to create 100 synthetic images with known geometric distortion. The discrepancies between known and measured GD was plotted in figure 4.14 with object nr. 23, the center point of the phantom, being assigned a zero distortion by definition. However, this is not the case as the measurement of this object will have a similar distribution of values as all other points and will add this uncertainty to that of each assessment of center-to-point distance. This is however included in the calculations and does not add any further spread to the values seen in figure 4.14. Note that the y-axis in this figure has the denotation pixels and that the largest errors made in estimation of GD is about 1 pixel. Depending on where in the image a single-pixel error is made the reported GD can vary significantly. Following the numbering system found in figure 3.9, object 16 will in the Philips large body phantom be 50 mm from the center, object 23. Assuming that there is a pixel size of $1 \times 1mm$, an error in object center placement of 1 pixel will be equivalent to an estimation error of 2 % geometric distortion. When taking into account the maximum accepted GD given by Philips in table 2.1, less than 1.4 % , it seems that this is at an excessive level. For objects further away from the center the effects of this error in object center estimation are much smaller. Following the same logic as before, object 1 is 180.28 mm away from the center object. This results in an estimation error of 0.55 % geometric distortion. It is therefore important to take object to center object distance into consideration when assessing the severity of GD in an image. It might be advisable to maintain the practise of only addressing the 20 outer most points in the image as is done in the current QA method, in accordance with NEMA standards. In figure 4.15 the distribution of measurement error in the geometric repetition test is given in percent. This shows a much larger variation in distribution that depends on the distance between phantom center and object of interest.

The mean value measured for each point in the phantom was anticipated to be zero and a two-sided t-test was performed to see if this hypothesis was rejected at the 5% significance level. It was not rejected, which shows that the mean of the difference between measured GD and known GD is not significantly different from zero.

The program does alleviate much of the work needed for the manual method, which MR-physicists at the hospital has called tedious and straining. But it also requires some insight and training to operate properly in cases where image contrast is poor, or objects in the periferal parts of the image get heavy distortions. One example of this is shown in figure 4.17, where the same DICOM image is analysed using two different options settings. Some images have low contrast in the periphery due to for example an inhomogenous magnetic field. When this problem occurs it becomes neccessary for the user to understand the effects of the relevant program settings to achieve a successful analysis. In figure 4.17c the values Sens and Edge have been changed from the default values to 0.96 and 0.2 respectively, and the result is that each measurement object is located. However, additional objects are also located and registered as possible measurement objects. The algorithms that handle these kinds of artifacts are only effective if there is sufficient difference between the artifact positions and the actual object positions, though this can also be adjusted by changing the dist_choice variable. The more artifacts that are discovered the longer the analysis tends to take, which is why it is advisable to keep variable values such as Sens and Edge to a working minimum.

5.2.2 SNR

From table 4.5 and the plot in figure **??** one can see that there are some discrepancies between measurements taken in 2012 using the IDL-based SNR program and in 2014 with the authors Matlab-based program (QAP). The discrepancy goes as high as $\pm$ 10 % , though only for a few measurements. This might be because of the method used in detecting the phantom. The IDL program uses a line profile technique technique described in section 3.6.1, while the QAP uses Hough transformation that is described in section 2.4.4. Combined with different variable choices the resulting circle will cover a slightly different area depending on which method is used

In addition to this there was discovered a slight variation in the method of data handling in the two different programs. The IDL program used a one dimensional vector approximation of the two dimensional ROI to represent the signal strength values inside the ROI, while the QAP used a two dimensional matrix built up using equation 2.11 and 2.12. This causes variation in the way ROI edge pixels are handled. It is at this point uncertain as to how much this could potentially contribute to a difference in SNR.

5.2.3 Uniformity

Table 4.6 shows the difference between new measurements of uniformity, taken with the Matlab program, and old measurements from the IDL program. This difference does not go above the value of 0.57 (unitless value), which can be argued to be a relatively small difference. There is no known value for the uniformity in these images. The method used for finding uniformity is very similar to the method used for finding spatial SNR, and the assignment of ROI is the same. Despite this the difference in measurement, percentagevise, is much smaller and more consistent.

5.2.4 Time series temporal stability

Since this is the first time that this method has been implemented for QA assessment at OUS there are no previous data available for comparison. Having a tool that automatically documents stability can be useful when undertaking studies that last a long period of time. It will be useful for tracking performance over several years, because it allows the user to build a library of previous measurements that can be plotted and easily read for early warnings. It may also be used for assessing drift in the scanner before use of a particular scanner, so that this may be corrected for during data analysis after a session. There is however room for much improvement, and at the time of writing there is no application involved that automatically pools results from several measurement sessions. Each session is currently kept in seperate files which increases manual effort and bookkeeping.

In Lee Friedman and Gary H. Glover: "Report on a multicenter fMRI quality assurance protocol" [7], the authors show that specific cites such as the University of Minnesota were able to detect and diagnose a malfunctioning head coil using a similar automated system for fMRI data. They also point out that these kinds of methods could be useful for studies that take long pauses between scans, and studies that are exposed to cite to cite differences in scanner performance.

5.3 Further development

5.3.1 IEC compliant scaling

The international electrotechnical commission (IEC) uses a different description of geometric distortion than the national electrical manufacturers association (NEMA) [5]. The concept of image scaling which is explained in section 2.4.4 is not thoroughly investigated in this thesis, but could be a natural extension to the existing methods used for detecting geometric distortion. Scaling is a seperate effect to GD, which affects an image equally in all directions resulting in a magnification or reduction effect of the whole object.

5.3.2 Weisskoff power spectrum analysis

In this thesis the background explaining Weisskoff analysis is shown in section 2.5.2. The program is however lacking when it comes to explaining the origins of the instabilities found. Weisskoff analysis is used to quantify the scanner instability found in functional MRI series. A power spectrum that is the absolute values of a fourier transformed time series, as used in the multicenter study of Friedman and Glover [7], will in the case of intervoxel correlation have a peak value that is statistically significant in relation to the surrounding noise in the frequency spectrum. It was thought that, if given enough time, an algorithm could be implemented to search for these kinds of peaks. This part of the thesis had to be postponed due to time constrains, but could perhaps be investigated in future projects by searching for periodic noise contributions that are caused by vibrations in the MR hardware.

Chapter 6

Conclusion

A fully functional quality assurance software package has been developed in this thesis. The program is designed to perform many of the same tasks that has previously been manual, or have been done in different software tools and packages within OUS. In addition to automate and rewrite previously used methods, several new methods have been included such as functional MR image analysis, Weisskoff analysis and temporal signal to noise ratio analysis.

It has been shown that an automated quality assurance protocol can produce results that are no less accurate than human expert evaluation, and do so using less time and manual effort. The amount of data to support the stability of this software is so far scarce and inconclusive, but shows early promise. The automatization enable instantanous printing of output to file for documentation, which earlier has been a relatively large part of the time consumption in manual analyses. The program will be integrated as a standard QA analysis at the MR research group at OUS, and will therefore quickly compile a large library of data for comparison with earlier analysis methods.

Spatial signal to noise ratio experiences some discrepancies with earlier methods, which is most likely due to the difference in the methods used in finding the phantom. The ROI is set to cover slightly different areas, which will therefore produce different SNR, due to the fact that the phantom images are not completely homogenous.

Chapter 7

Matlab Code

This section contains most of the code developed in this thesis, with the exeption of the graphic user interface code (GUI), which mostly consists of generated code from the GUIDE auxiliary program. This code is heavy on comments and is not neccessary in order to understand each seperate program in the package. It is useful however to keep in mind that it is in the GUI code that each program is tied together, and a few variables are defined there as well.

7.1 Geometric distortion

7.1.1 FindCircles.m

```
1  % Robin Antony Birkeland Bugge. November 2013
2  % Find circles
3  % uses the imfindcircles function
4  % ************************************************************************
5
6
7  Edge = 0.3;
8  Sens = 0.9;
9  r_min = 2;
10 r_max = 8;
11
12 if ~isempty(get(handles.edit_Sens,'String'))
13     Sens = get(handles.edit_Sens,'String');
14     Sens = str2double(Sens);
15 end %if
16
17 if ~isempty(get(handles.edit_Edge,'String'))
18     Edge = get(handles.edit_Edge,'String');
19     Edge = str2double(Edge);
20 end %if
```

```
21
22 if ~isempty(get(handles.edit_r_min,'String'))
23     r_min = get(handles.edit_r_min,'String');
24     r_min = str2double(r_min);
25 end %if
26
27 if ~isempty(get(handles.edit_r_max,'String'))
28     r_max = get(handles.edit_r_max,'String');
29     r_max = str2double(r_max);
30 end %if
31
32 [centers_bright, radii_bright] = imfindcircles(img,[r_min r_max],'
       ObjectPolarity','bright', ...
33     'Sensitivity',Sens,'EdgeThreshold',Edge);
34
35 numberOfCircles = size(centers_bright,1);
36
37 % Object continuation artifacts
38 % *****************************************************************
39
40 art_min = 20; %this sets the minimum distance between objects
41 artifact_flag = 0;
42 if ~isempty(get(handles.edit_art_min,'String'))
43     art_min = get(handles.edit_art_min,'String');
44     art_min = str2double(art_min);
45 end %if
46
47 o = 1;
48 cont_art_min = 8;
49
50 % Artifact remover. Locates phantom holding structure
51 while o <= numberOfCircles && numberOfCircles > nr_of_objects
52 artifact_flag = 0;
53
54 current_x = round(centers_bright(o,1));
55 current_y = round(centers_bright(o,2));
56 signal_max = impixel(img,current_x, current_y);
57
58 for i = 1:double(Tags.Rows)
59     for j = 1:double(Tags.Columns)
60
61         L = sqrt((current_x - i)^2 + (current_y - j)^2);
62
63         if L == cont_art_min
64             signal = impixel(img,i,j);
65             if signal(1) >= signal_max(1)*0.3
66
67                 centers_bright(o,:) = [];
68                 radii_bright(o) = [];
69                 artifact_flag = 1;
70             end %if
71         end %if
72         if artifact_flag == 1
73            break
74         end %if
75     end %for
76     if artifact_flag == 1
77        break
78     end %if
79 end %for
80
81 figure(1)
```

```
imshow(img,'DisplayRange',[])
h_bright = viscircles(centers_bright,radii_bright);

if artifact_flag == 0
   o = o+1
end %if
numberOfCircles = size(centers_bright,1);
end %while
o = 1;
% ************************************************************

figure(1)
imshow(img,'DisplayRange',[])
h_bright = viscircles(centers_bright,radii_bright);
%h_dark = viscircles(centers_dark,radii_dark,'EdgeColor','b');
%h_Phantom = viscircles(Phantom_center, Phantom_radii);

error_indx = [];
i = 1;
d = zeros(1,numberOfCircles);
err_matrix_pos = 1;
l = zeros(1,numberOfCircles);
d = zeros(1,numberOfCircles);

numberOfCircles = size(centers_bright,1);

[x,y] = ginput(1);
```

7.1.2 Geodist_reader.m

```
% Robin Antony Birkeland Bugge
% discom reader for geodist program
% ************************************************************************



[Filename,PathName,FilterIndex] = uigetfile('*.*');
img_address = strcat(PathName,Filename);

img = dicomread(img_address);
```

7.1.3 geometric2D.m

```
% Robin Antony Birkeland Bugge
% 2D geodist script
% ************************************************************************

%function geometric2D


warning('off','all'); % Turns off warnings. This is used because of
% limitations in the imfindcircles function. Change parameter 'off' to
% 'on' in order to view warnings


val = get(handles.phantom_list,'Value');

switch val
    case 1
```

```
17          nr_of_objects = 45; %phillips_positions_manual
18      case 2
19          nr_of_objects = 45; %phillips_small_neuronal_positions_manual
20  end %switch
21
22
23  % Read in important tags and write to textfile.
24  % ************************************************************************
25
26  TagPrinter
27
28  % Initiate flags
29  % ************************************************************************
30
31  fiveface = 1;
32  quadrant = 0;
33
34
35  % imfindcircles, image processing toolbox
36  % ************************************************************************
37
38  FindCircles
39
40
41  % Calculates distance to approximate center and tests for artifacts
42  % Artifact removal procedure
43  % This will attempt to find objects that are too closely spaced
44  % Includes a configurable minimum distance
45
46  max_err = 3; % maximum number of errors before artifacts assumption
47
48  if ~isempty(get(handles.edit_max_err,'String'))
49      max_err = get(handles.edit_max_err,'String');
50      max_err = str2double(max_err);
51  end %if
52
53  % Artifact detectors
54  for i = 1:numberOfCircles
55
56      err = 0;
57
58      current_xpos = centers_bright(i,1);
59      current_ypos = centers_bright(i,2);
60
61      current_dist = sqrt(((current_xpos - x)^2) + (current_ypos - y)^2);
62
63      for j = 1:numberOfCircles
64
65          current_xpos_j = centers_bright(j,1);
66          current_ypos_j = centers_bright(j,2);
67
68          artifact_dist = sqrt(((current_xpos - current_xpos_j)^2) + ...
69          (current_ypos - current_ypos_j)^2);
70
71          l(1,j) = artifact_dist;
72
73          if l(1,j) < art_min
74              err = err +1;
75          end % if
76
77
78      end % for
```

```
79
80      if err < max_err
81
82          %hold on
83          %plot(current_xpos,current_ypos,'o r')
84          %hold off
85
86          d(1,i) = current_dist; % exports information about distance
87
88      else
89
90          error_indx(err_matrix_pos) = i;
91          err_matrix_pos = err_matrix_pos +1;
92          d(1,i) = 999;
93
94
95      end % if
96
97  end %for
98
99
100 %Finds the objects based on index numbers
101
102 clone = d;
103
104 [d1,indx1] = min(d);
105 length(1) = d1;
106 d(indx1) = 999999;
107
108 [d2,indx2] = min(d);
109 length(2) = d2;
110 d(indx2) = 999999;
111
112 [d3,indx3] = min(d);
113 length(3) = d3;
114 d(indx3) = 999999;
115
116 [d4,indx4] = min(d);
117 length(4) = d4;
118 d(indx4) = 999999;
119
120 [d5,indx5] = min(d);
121 length(5) = d5;
122 d(indx5) = 999999;
123
124 d = clone;
125
126 % Quadrant method
127 if quadrant == 1
128
129 indx = [indx1 indx2 indx3 indx4];
130 indx_sort = sort(indx);
131
132 first = indx_sort(1);
133 second = indx_sort(2);
134 third = indx_sort(4);
135 fourth = indx_sort(3);
136
137 x1 = centers_bright(first,1);
138 y1 = centers_bright(first,2);
139
140 x2 = centers_bright(second,1);
```

```
141 y2 = centers_bright(second,2);
142
143 x3 = centers_bright(third,1);
144 y3 = centers_bright(third,2);
145
146 x4 = centers_bright(fourth,1);
147 y4 = centers_bright(fourth,2);
148
149 % Finds the center of phantom using quadrants
150
151
152 hold on
153 line([x1 x4],[y1 y4])
154 line([x2 x3],[y2 y3])
155 hold off
156
157 [xc,yc] = polyxpoly([x1 x4],[y1 y4],[x2 x3],[y2 y3]); % Center position
158
159 end %if
160
161 % Five face method
162
163 if fiveface == 1
164
165         center_indx = indx1;
166         xc = centers_bright(center_indx,1);
167         yc = centers_bright(center_indx,2);
168
169         hold on
170         plot(xc,yc,'x')
171         hold off
172
173 end % if
174
175
176 % Creates matrices and vectors for positioning of the undistorted phantom
177 % ************************************************************************
178
179 val = get(handles.phantom_list,'Value');
180
181 switch val
182     case 1
183         phillips_positions_manual
184     case 2
185         phillips_small_neuronal_positions_manual
186 end %switch
187
188 % Object chooser, automatic input. Uses all objects in plane
189 % ************************************************************************
190
191 % Isocenter position
192 if isfield(info,'ImagePositionsPatient')
193
194     Ix = abs(info.ImagePositionPatient(1));
195     Iy = abs(info.ImagePositionPatient(2));
196     Iz = abs(info.ImagePositionPatient(3));
197
198 else
199     % Setter isocenter til sentrum av fantomet
200     Ix = xc;
201     Iy = yc;
202     Iz = 0;
```

```
203
204 end %if
205
206 dist_choice = 30;
207
208 if ~isempty(get(handles.edit_dist,'String'))
209     dist_choice = get(handles.edit_dist,'String');
210     dist_choice = str2double(dist_choice);
211 end %if
212
213 dist_threshold = dist_choice*scale;
214
215 counter = 1;
216 s = size(positions);
217
218 Da = zeros(s(1),1);
219 Dm = zeros(s(1),1);
220 sorted = zeros(s(1),2);
221 dist_vec = zeros(s(1),1);
222
223 center = sortrows(centers_bright,1);
224 x_dist = zeros(numberOfCircles,1);
225 y_dist = zeros(numberOfCircles,1);
226
227 for i = 1:s(1)
228
229     current_x = positions(i,1);
230     current_y = positions(i,2);
231
232     for j = 1:numberOfCircles
233
234         dist = sqrt((current_x - center(j,1))^2+(current_y - center(j,2))^2)
                ;
235
236
237         if dist <= dist_threshold
238
239             sorted(counter,1) = center(j,1);
240             sorted(counter,2) = center(j,2);
241
242             dist_vec(i) = dist;
243
244             Dm(counter) = sqrt((xc - sorted(counter,1))^2 + (yc - sorted(
                    counter,2))^2);
245             Da(counter) = sqrt((current_x - xc)^2 + (current_y - yc)^2);
246             counter = counter + 1;
247
248             x_dist(j) = positions(i,1) - center(j,1);
249             y_dist(j) = positions(i,2) - center(j,2);
250
251             %add blue marker
252             hold on
253             plot(center(j,1),center(j,2),'x')
254             hold off
255
256             break
257         end %if
258     end %for
259
260 end %for
261
262 difference = Dm - Da;
```

```
263 diff = max(abs(differance));
264
265 % Scatterplot over distances
266 % ************************************************************************
267
268
269 dx = differance; % Vector for differance in position. a.k.a distortion
270 drc = sqrt((xc - Ix)^2 + (yc - Iy)^2 + (0 - Iz)^2);
271
272 S = size(Da);
273 Di = zeros(1,S(1));
274
275 v1 = [Ix,Iy];
276 for i = 1:S(1)
277
278 % v2 = [positions(i,1),positions(i,2)];
279 % prod = drc*Da(i);
280
281 Di(i) = sqrt((positions(i,1)-Ix)^2 + (positions(i,2)-Iy)^2); %isocenter
      lengths
282
283 end
284
285 FoV = double(FoV);
286
287 dist_mm = Dm*(FoV/Standard_row);
288 mm = dist_mm' - Di;
289
290 x_dist_mm = x_dist*(FoV/Standard_row);
291 y_dist_mm = y_dist*(FoV/Standard_row);
292
293 percent = (mm./Di)*100;
294
295 [max_percent,i] = max(abs(percent));
296 GD = max_percent; % Maximum geometric distortion in percentage
297
298 % Sort from highest to lowest
299 ID = linspace(1,s(1),s(1));
300 D = [ID;Di;Dm';dist_mm';dx';mm;percent;positions(:,1)';positions(:,2)';
      x_dist';y_dist';x_dist_mm';y_dist_mm'];
301 D = D';
302 D = sortrows(D,1);
303
304 Di = D(:,2);
305 dx = D(:,4);
306 ID = D(:,1);
307
308 % plot
309
310 Plotter
311
312
313 %write neccessary info onto the file
314 % ************************************************************************
315
316 info_printer
317
318 % ************************************************************************
319
320
321
322 %end
```

7.1.4 info_printer.m

```
% Robin Antony Birkeland Bugge. November 2013
% info_printer
% ************************************************************************


% Maximum differance in mm
fprintf(fid,'%s:','Maximum differance in distance (mm)');
fprintf(fid,'%6.2f:',max(abs(mm)));
fprintf(fid,'%6.2f\n',find(abs(mm)==max(abs(mm))));


% Maximum geometric distortion in percentage
fprintf(fid,'%s:','Maximum geometric distortion (percentage)');
fprintf(fid,'%6.2f:',max(abs(percent)));
fprintf(fid,'%6.2f\n',find(abs(percent)==max(abs(percent))));


% Spacer
fprintf(fid,'\n');

% Print header for info
fprintf(fid,'%s, \n','Object nr,Original distance (pxl),Measured distance (
    pxl), Distance (mm),dl (pxl),dl (mm),Percentage,x,y,dx (pxl), dy (pxl),
    dx (mm), dy (mm)');

% Prints the matrix D that contains info on each object
for i = 1:size(D,1)
    fprintf(fid, '%f,', D(i,:));
    fprintf(fid,'\n');
end %for

% Output for GUI

val = num2str(diff);
set(handles.max_pixel_out,'String', val);

val = num2str(GD);
set(handles.max_percentage_out,'String', val);

val = size(center);
val = val(1);
set(handles.num_obj_out,'String', val);
```

7.1.5 phillips_positions_manual.m

```
% Robin Antony Birkeland Bugge. October 2013
% Vector with phillips phantom positions.

% ************************************************************************

Standard_row = 448;
Standard_column = 448;

current_row = double(Tags.Rows);
if isfield(info,'PixelSpacing')
    FoV = double(Tags.Rows) * double(info.PixelSpacing(1));
else
    FoV = double(Tags.Rows);
```

```
14 end
15 % Assumes equal row and column length
16 %s = current_row/Standard_row;
17 s = 1;
18 % Larger positions
19 positions = [xc - 3*50*s yc - 2*50*s; xc - 3*50*s yc - 1*50*s; xc - 3*50*s
       yc; xc - 3*50*s yc + 50*s; xc - 3*50*s yc + 2*50*s; ...
20              xc - 2*50*s yc - 3*50*s; xc - 2*50*s yc - 2*50*s; xc - 2*50*s
                    yc - 1*50*s; xc - 2*50*s yc; xc - 2*50*s yc + 50*s; xc -
                    2*50*s yc + 2*50*s; xc - 2*50*s yc + 3*50*s; ...
21              xc - 1*50*s yc - 3*50*s; xc - 1*50*s yc - 2*50*s; xc - 1*50*s
                    yc - 1*50*s; xc - 1*50*s yc; xc - 1*50*s yc + 50*s; xc -
                    1*50*s yc + 2*50*s; xc - 1*50*s yc + 3*50*s; ...
22              xc         yc - 3*50*s; xc         yc - 2*50*s; xc         yc -
                    1*50*s; xc         yc; xc         yc + 50*s; xc         yc +
                    2*50*s; xc         yc + 3*50*s; ...
23              xc + 1*50*s yc - 3*50*s; xc + 1*50*s yc - 2*50*s; xc + 1*50*s
                    yc - 1*50*s; xc + 1*50*s yc; xc + 1*50*s yc + 50*s; xc +
                    1*50*s yc + 2*50*s; xc + 1*50*s yc + 3*50*s; ...
24              xc + 2*50*s yc - 3*50*s; xc + 2*50*s yc - 2*50*s; xc + 2*50*s
                    yc - 1*50*s; xc + 2*50*s yc; xc + 2*50*s yc + 50*s; xc +
                    2*50*s yc + 2*50*s; xc + 2*50*s yc + 3*50*s; ...
25              xc + 3*50*s yc - 2*50*s; xc + 3*50*s yc - 1*50*s; xc + 3*50*s
                    yc; xc + 3*50*s yc + 50*s; xc + 3*50*s yc + 2*50*s];
26
27
28 scale = s;
29 nr_columns = 7; %this should be automated
```

7.1.6 phillips_small_neuronal_positions_manual.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Vector with phillips phantom positions.
3
4  % ************************************************************************
5
6  Standard_row = 256;
7  Standard_column = 256;
8
9  current_row = double(Tags.Rows);
10 FoV = FieldOfView;
11 % Assumes equal row and column length
12
13 s = 1;
14 % Larger positions
15 positions = [xc - 3*25*s yc - 2*25*s; xc - 3*25*s yc - 1*25*s; xc - 3*25*s
       yc; xc - 3*25*s yc + 25*s; xc - 3*25*s yc + 2*25*s; ...
16              xc - 2*25*s yc - 3*25*s; xc - 2*25*s yc - 2*25*s; xc - 2*25*s
                    yc - 1*25*s; xc - 2*25*s yc; xc - 2*25*s yc + 25*s; xc -
                    2*25*s yc + 2*25*s; xc - 2*25*s yc + 3*25*s; ...
17              xc - 1*25*s yc - 3*25*s; xc - 1*25*s yc - 2*25*s; xc - 1*25*s
                    yc - 1*25*s; xc - 1*25*s yc; xc - 1*25*s yc + 25*s; xc -
                    1*25*s yc + 2*25*s; xc - 1*25*s yc + 3*25*s; ...
18              xc         yc - 3*25*s; xc         yc - 2*25*s; xc         yc -
                    1*25*s; xc         yc; xc         yc + 25*s; xc         yc +
                    2*25*s; xc         yc + 3*25*s; ...
19              xc + 1*25*s yc - 3*25*s; xc + 1*25*s yc - 2*25*s; xc + 1*25*s
                    yc - 1*25*s; xc + 1*25*s yc; xc + 1*25*s yc + 25*s; xc +
                    1*25*s yc + 2*25*s; xc + 1*25*s yc + 3*25*s; ...
20              xc + 2*25*s yc - 3*25*s; xc + 2*25*s yc - 2*25*s; xc + 2*25*s
                    yc - 1*25*s; xc + 2*25*s yc; xc + 2*25*s yc + 25*s; xc +
```

```
                2*25*s yc + 2*25*s; xc + 2*25*s yc + 3*25*s; ...
21              xc + 3*25*s yc - 2*25*s; xc + 3*25*s yc - 1*25*s; xc + 3*25*s
                yc; xc + 3*25*s yc + 25*s; xc + 3*25*s yc + 2*25*s];
22
23
24  scale = s;
25  nr_columns = 7; %this should be automated
```

7.1.7 Plotter.m

```
1   % Robin Antony Birkeland Bugge. November 2013
2   % Plotter. Highly specific for the Philips phantom
3   % ************************************************************************
4
5
6   zero = zeros(1,S(1));
7
8   figure(2)
9
10  hold on
11  for i = 1:size(dx,1)
12
13      if ID(i) <= 5
14
15          plot(Di(i),dx(i),'ko')
16          hold on
17      end
18      if ID(i) >= 6 && ID(i) <= 12
19
20          plot(Di(i),dx(i),'ro')
21          hold on
22      end
23      if ID(i) >= 13 && ID(i) <= 19
24
25          plot(Di(i),dx(i),'yo')
26          hold on
27      end
28      if ID(i) >= 20 && ID(i) <= 26
29
30          plot(Di(i),dx(i),'go')
31          hold on
32      end
33      if ID(i) >= 27 && ID(i) <= 33
34
35          plot(Di(i),dx(i),'mo')
36          hold on
37      end
38      if ID(i) >= 34 && ID(i) <= 40
39
40          plot(Di(i),dx(i),'co')
41          hold on
42      end
43      if ID(i) >= 41 && ID(i) <= 45
44
45          plot(Di(i),dx(i),'bo')
46          hold on
47      end
48
49  end % for
50
```

```
51  plot(Di,zero,'r')
52  title('Scatterplot, Distance from isocenter vs. measurement differentials')
53  xlabel('Di (Distance from isocenter)')
54  ylabel('dx (Difference between measured and known positions)')
55  axis([0 (max(Di)+0.1*max(Di)) (min(dx)-(0.1*abs(min(dx)))) (max(dx)+(0.1*abs
        (max(dx))))])
56
57  hold off
58
59  figure(5)
60  hist(percent)
61  xlabel('Percentage distortion')
62  ylabel('Number of objects')
63
64  % Meshgrid
65
66  mesh_x = [-3 -2 -1 0 1 2 3];
67  mesh_y = [-3 -2 -1 0 1 2 3];
68  mesh_dx = [nan mm(1) mm(2) mm(3) mm(4) mm(5) nan; ...
69             mm(6) mm(7) mm(8) mm(9) mm(10) mm(11) mm(12); ...
70             mm(13) mm(14) mm(15) mm(16) mm(17) mm(18) mm(19); ...
71             mm(20) mm(21) mm(22) mm(23) mm(24) mm(25) mm(26); ...
72             mm(27) mm(28) mm(29) mm(30) mm(31) mm(32) mm(33); ...
73             mm(34) mm(35) mm(36) mm(37) mm(38) mm(39) mm(40); ...
74              nan mm(41) mm(42) mm(43) mm(44) mm(45) nan];
75
76  figure(6)
77  surf(mesh_x,mesh_y,mesh_dx)
78  xlabel('y')
79  ylabel('x')
80  zlabel('Distortion distance (mm)')
```

7.1.8 TagPrinter.m

```
1  % Robin Antony Birkeland Bugge. November 2013
2  % Tag printer
3  % ************************************************************************
4
5  [FileName,Pathname] = uiputfile('*.xls');
6  info = dicominfo(img_address);
7  file = strcat(Pathname,'\',FileName);
8  fid = fopen(file,'wt');
9
10 fprintf(fid,'%s\n',' ');
11
12 if isfield(info,'ImagePositionPatient') == 1
13
14     Tags.ImagePositionPatient = info.ImagePositionPatient;
15     Field = 'ImagePositionPatient';
16
17     fprintf(fid,'%s:',Field);
18     fprintf(fid,'%6.2f,%6.2f,%6.2f\n',Tags.ImagePositionPatient);
19
20 end %if
21
22 if isfield(info,'SeriesDescription') == 1
23
24     Tags.SeriesDescription = info.SeriesDescription;
25     Field = 'SeriesDescription';
26
```

```
27      fprintf(fid,'%s:',Field);
28      fprintf(fid,'%s\n',Tags.SeriesDescription);
29
30  end %if
31
32  if isfield(info,'ImageOrientationPatient') == 1
33      Tags.ImageOrientationPatient = info.ImageOrientationPatient;
34      Field = 'ImageOrientationPatient';
35
36      if info.ImageOrientationPatient(1) == 1 && abs(info.
            ImageOrientationPatient(6)) == 1
37          % Kor
38          fprintf(fid,'%s:',Field);
39          fprintf(fid,'%s\n','Kor');
40      end
41
42      if info.ImageOrientationPatient(1) == 1 && abs(info.
            ImageOrientationPatient(5)) == 1
43          % Tra
44          fprintf(fid,'%s:',Field);
45          fprintf(fid,'%s\n','Tra');
46      end
47
48      if info.ImageOrientationPatient(2) == 1 && abs(info.
            ImageOrientationPatient(6)) == 1
49          % Sag
50          fprintf(fid,'%s:',Field);
51          fprintf(fid,'%s\n','Sag');
52      end
53
54  end %if
55
56  if isfield(info,'Rows') == 1
57      Tags.Rows = info.Rows;
58      Field = 'Rows';
59
60      fprintf(fid,'%s:',Field);
61      fprintf(fid,'%6.2f\n',Tags.Rows);
62
63      if isfield(info,'PixelSpacing') == 1
64          fprintf(fid,'%s:','FieldOfView_Rows');
65          FieldOfView_row = Tags.Rows * info.PixelSpacing(1);
66          fprintf(fid,'%6.2f\n',FieldOfView_row);
67      end %if
68
69
70  end %if
71
72  if isfield(info,'Columns') == 1
73      Tags.Columns = info.Columns;
74      Field = 'Columns';
75
76      fprintf(fid,'%s:',Field);
77      fprintf(fid,'%6.2f\n',Tags.Columns);
78
79      if isfield(info,'PixelSpacing') == 1
80          fprintf(fid,'%s:','FieldOfView_Columns');
81          FieldOfView_column = Tags.Columns * info.PixelSpacing(1);
82          fprintf(fid,'%6.2f\n',FieldOfView_column);
83      end %if
84
85  end %if
```

```
86
87  if isfield(info,'PixelSpacing') == 1
88      Tags.PixelSpacing = info.PixelSpacing;
89      Field = 'PixelSpacing';
90
91      fprintf(fid,'%s:',Field);
92      fprintf(fid,'%6.2f,%6.2f\n',Tags.PixelSpacing);
93
94  end %if
95
96  if isfield(info,'AcquisitionDate') == 1
97      Tags.AcquisitionDate = info.AcquisitionDate;
98      Field = 'AcquisitionDate';
99
100     fprintf(fid,'%s:',Field);
101     fprintf(fid,'%s\n',Tags.AcquisitionDate);
102
103 end %if
104
105 if isfield(info,'AcquisitionTime') == 1
106     Tags.AcquisitionTime = info.AcquisitionTime;
107     Field = 'AcquisitionTime';
108
109     fprintf(fid,'%s:',Field);
110     fprintf(fid,'%s\n',Tags.AcquisitionTime);
111
112 end %if
113
114 if isfield(info,'Manufacturer') == 1
115     Tags.Manufacturer = info.Manufacturer;
116     Field = 'Manufacturer';
117
118     fprintf(fid,'%s:',Field);
119     fprintf(fid,'%s\n',Tags.Manufacturer);
120
121 end %if
122
123 if isfield(info,'InstitutionName') == 1
124     Tags.InstitutionName = info.InstitutionName;
125     Field = 'InstitutionName';
126
127     fprintf(fid,'%s:',Field);
128     fprintf(fid,'%s\n',Tags.InstitutionName);
129
130 end %if
131
132 if isfield(info,'StationName') == 1
133     Tags.StationName = info.StationName;
134     Field = 'StationName';
135
136     fprintf(fid,'%s:',Field);
137     fprintf(fid,'%s\n',Tags.StationName);
138
139 end %if
140
141 if isfield(info,'SliceThickness') == 1
142     Tags.SliceThickness = info.SliceThickness;
143     Field = 'SliceThickness';
144
145     fprintf(fid,'%s:',Field);
146     fprintf(fid,'%6.2f\n',Tags.SliceThickness);
147
```

```
148 end %if
149
150 seperator = '
        ********************************************************************************
        ';
151 fprintf(fid,'%s\n','');
152 fprintf(fid,'%s:',seperator);
153 fprintf(fid,'%s:','Value');
154 fprintf(fid,'%s\n','Object number');
```

7.2 Time series

7.2.1 Analyzer.m

```
1  % Robin Birkeland Bugge. November 2013
2  % Analyzer. Make statistical assessments and Weisskoff
3  % ************************************************************************
4
5
6  % Weisskoff. Uses different ROI's
7  % ************************************************************************
8
9  signal_number = r_end - r_start + 1;
10 SD = zeros(signal_number,1);
11 mean_signal = zeros(signal_number,1);
12 CV = zeros(signal_number,1);
13 ROI = linspace(r_start,r_end,signal_number);
14
15 for i = 1:r_end - r_start + 1
16 SD(i) = std(signal(:,i));
17 mean_signal(i) = mean(signal(:,1));
18
19 CV(i) = (SD(i)/mean_signal(i)) * 100;
20 end % for
21
22 theo_CV = 1./sqrt(ROI); % Theoretical CV
23
24 figure(2)
25 plot(log(ROI),log(CV),'o')
26 hold on
27 plot(log(ROI),log(theo_CV),'ro')
28 hold off
29 xlabel('ROI radii');
30 ylabel('CV');
31 title('Measured and theoretical CV vs. ROI radii');
32
33 % ************************************************************************
34
35 % Values of interest
36
37 poly_values = polyval(p,l);
38
39
40 % Evaluate residuals for signal vector created with ROI of "midle" size
41 residual = poly_values - signal(:,half_ROI);
42
43 res_SD = std(residual);
44
```

```
45 % Percent fluctuation
46 PF = 100*(res_SD/mean(signal(:,half_ROI)));
47
48 set(handles.edit_PF,'String', num2str(PF));
49
50
51 % Drift value
52 drift_value = 100*(max(poly_values)-min(poly_values))/mean(signal(:,half_ROI
       ));
53
54 set(handles.edit_drift,'String', num2str(drift_value));
55
56
57 % Combined matrix
58
59 M = [signal(:,half_ROI) residual];
60
61 % mean, max and min signal
62
63 set(handles.edit_avg_int,'String', num2str(mean(signal(:,half_ROI))));
64 set(handles.edit_min_int,'String', num2str(min(signal(:,half_ROI))));
65 set(handles.edit_max_int,'String', num2str(max(signal(:,half_ROI))));
66
67
68 %SNR analysis
```

7.2.2 Drift.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Drift analysis
3  % ************************************************************************
4
5  start = 1;
6  n = 100;
7
8  r_start = 1;
9  r_end = 20;
10
11 R = 10;
12
13 if ~isempty(get(handles.edit_start,'String'))
14     start = get(handles.edit_start,'String');
15     start = str2double(start);
16 end %if
17
18 if ~isempty(get(handles.edit_n,'String'))
19     n = get(handles.edit_n,'String');
20     n = str2double(n);
21 end %if
22
23 if ~isempty(get(handles.edit_r_start,'String'))
24     r_start = get(handles.edit_r_start,'String');
25     r_start = str2double(r_start);
26 end %if
27
28 if ~isempty(get(handles.edit_r_end,'String'))
29     r_end = get(handles.edit_r_end,'String');
30     r_end = str2double(r_end);
31 end %if
32
```

```
33 if ~isempty(get(handles.edit_R,'String'))
34     R = get(handles.edit_R,'String');
35     R = str2double(R);
36 end %if
37
38 handles.start = start;
39 handles.n = n;
40 handles.r_start = r_start;
41 handles.r_end = r_end;
42 guidata(hObject,handles);
43
44 % prompt = 'Do you wish to save? (y/n): ';
45 % save_flag = input(prompt,'s');
46 %
47 % if save_flag == 'y'
48 %
49 %     [FileName,Pathname] = uiputfile('*.xls');
50 % end %if
51
52 Drift_ROI_images
53
54 Polyfitter % Fits a curve to the data
55
56 Analyzer % Creates statistics and Weisskoff analysis
57
58
59 % Plot
60 % figure(2)
61 %
62 % plot(l,signal(:,half_ROI),'*',start:0.1:length(l),polyval(p,start:0.1:
       length(l)),'-')
63 % hold on
64 % plot(l,signal(:,half_ROI),'r')
65 % xlabel('Image number')
66 % ylabel('Average signal')
67 % titl = strcat('Variable ROI = [',num2str(r_start),',',num2str(r_end),']');
68 % title(titl)
69
70 hold off
71
72
73 figure(3)
74 plot(l,signalR(:,counter1),'*',start:0.1:length(l),polyval(pR,start:0.1:
       length(l)),'-')
75 hold on
76 plot(l,signalR(:,counter1),'r')
77 xlabel('Image number')
78 ylabel('Average signal')
79 titl = strcat('ROI = ',num2str(R));
80 title(titl)
81 % if save_flag == 'y'
82 % Drift_printer
83 % end %if
```

7.2.3 Drift_printer.m

```
1 % Robin Antony Birkeland Bugge. October 2013
2 % info printer til excel format
3 % ************************************************************************
4
```

```
file = strcat(Pathname,'\',FileName);
fid = fopen(file,'wt');


for i = 1:size(M,1)
    fprintf(fid, '%f,', M(i,:));
    fprintf(fid,'\n');
end %for


fclose(fid);
```

7.2.4 Drift_reader.m

```
% Robin Antony Birkeland Bugge. August 2013.
% Dicomleser for Drift.m
% ************************************************************************

[path] = uigetdir;
val = strcat(path,'\','*.dcm');

handles.path = path;
guidata(hObject,handles);
```

7.2.5 Drift_ROI_images.m

```
% Robin Antony Birkeland Bugge. October 2013
% Drift image counter
% ************************************************************************

if isfield(handles,'SNR') == 1
    SNR_switch = get(handles.SNR_switch,'String');
    SNR_switch = str2double(SNR_switch);
else
    SNR_switch = 0;
end %if

path = handles.path;

val = strcat(path,'\','*.dcm');
fnames = dir(val);
nr = length(fnames);
nr_img = nr;
nrR = length(fnames);
%half_point = nr/2;

sum = 0;
signal = zeros(nr_img,r_end-r_start+1);
signalR = zeros(nr_img,r_end-r_start+1);
signal_sum = 0;
signal_sumR = 0;
flag = 0;
l = linspace(start,nr_img,nr_img);
counter1 = 0;

signal_SNR1 = 0;
signal_SNR2 = 0;

Sens = 0.9;
```

```
34  Edge = 0.3;
35
36  r_small = 25;
37  if ~isempty(get(handles.edit_r_small,'String'))
38      r_small = get(handles.edit_r_small,'String');
39      r_small = str2double(r_small);
40  end %if
41
42  r_large = 45;
43  if ~isempty(get(handles.edit_r_large,'String'))
44      r_large = get(handles.edit_r_large,'String');
45      r_large = str2double(r_large);
46  end %if
47
48  warning('off','all');
49
50  % ************************************************************************
51
52  % Les inn og average over ROI
53  % ************************************************************************
54
55  % img = dicomread(strcat(path,'\',fnames(1,1).name));
56  % dim = size(img);
57  % Matrix = zeros(dim(1),2);
58  % for i = 1:dim(2)
59  %     for j = 1:dim(1)
60  %         L = sqrt((i - xc)^2 + (j - yc)^2);
61  %         if L <= r
62  %             Matrix(i,j) = 1;
63  %         else
64  %             Matrix(i,j) = 0;
65  %         end %if
66  %     end %for
67  % end %for
68  %
69  % Matrix = double(Matrix);
70  %
71  % img_matrix = img.*Matrix;
72  %
73  % imshow(img_matrix)
74
75  for r = r_start:r_end
76      counter1 = counter1 + 1
77  for i = start:nr_img
78
79      nr = 0;
80      nrR = 0;
81      name_nr = strcat('Image',num2str(i));
82      Images.(name_nr) = dicomread(strcat(path,'\',fnames(i,1).name));
83      Info.(name_nr) = dicominfo(strcat(path,'\',fnames(i,1).name));
84      dim = size(Images.(name_nr));
85      signal_sum = 0;
86      signal_sumR = 0;
87      signal_sum_SNR = 0;
88      img = double(Images.(name_nr));
89      s_a = zeros(dim(2),1);
90      s_a_SNR = zeros(dim(2),1);
91
92      if i == start
93          SNR_vec = zeros(dim(1),dim(2));
94      end %if
95
```

```
96      if i == start
97      if SNR_switch == 1
98          SNR
99      end %if
100     end %if
101
102     if flag == 0
103         figure(1)
104         [centers_bright, radii_bright] = imfindcircles(img,[r_small r_large
                ],'ObjectPolarity','bright', ...
105         'Sensitivity',Sens,'EdgeThreshold',Edge);
106
107         imshow(img,'DisplayRange',[],'Parent',handles.axes1);
108         viscircles(centers_bright,radii_bright);
109
110         % find center
111         xc = centers_bright(1);
112         yc = centers_bright(2);
113
114         centers = [xc,yc];
115
116         hold on
117         plot(xc,yc,'x')
118         hold off
119
120         flag = 1;
121     end
122
123     figure(1)
124     imshow(Images.(name_nr),'DisplayRange',[])
125     viscircles(centers, r)
126     viscircles(centers_bright,radii_bright);
127     hold on
128     plot(xc,yc,'x')
129     hold off
130
131     for x = 1:dim(1)
132
133         for y = 1:dim(2)
134
135             current_signal = img(x,y);
136
137             % SNR image signal
138             if SNR_switch == 1 && i == 1
139                 current_signal_SNR = diff_img(x,y);
140             end %if
141
142             %current_signal_array(x,y) = current_signal;
143
144             dist = sqrt((xc - x)^2+(yc - y)^2);
145
146             if dist <= r
147
148                 signal_sum = signal_sum + current_signal;
149                 s_a(y) = signal_sum;
150                 nr = nr + 1;
151
152             end %if
153
154             if dist <= R
155
156                 signal_sumR = signal_sumR + current_signal;
```

```
157                     s_a(y) = signal_sumR;
158                     nrR = nrR + 1;
159
160                     if SNR_switch == 1 && i == 1
161                         SNR_vec(x,y) = current_signal_SNR;
162                         signal_sum_SNR = signal_sum_SNR + current_signal_SNR;
163                         s_a_SNR(y) = signal_sum_SNR;
164                     end %if
165
166                 end %if
167
168             end %for
169
170         end %for
171
172         signal(i-start + 1,counter1) = signal_sum/nr;
173         signalR(i-start + 1,counter1) = signal_sumR/nrR;
174
175
176
177 end %for
178 end %for
```

7.2.6 SNR.m

```
1  % Robin Antony Birkeland Bugge. February 2014
2  % SNR setup for drift module
3  % ****************************************
4
5  if ~isempty(get(handles.edit_Image1,'String'))
6      img_nr1 = get(handles.edit_Image1,'String');
7      img_nr1 = str2double(img_nr1);
8  else
9      img_nr1 = nr_img/2;
10 end %if
11
12 if ~isempty(get(handles.edit_Image2,'String'))
13     img_nr2 = get(handles.edit_Image2,'Double');
14     img_nr2 = str2double(img_nr2);
15 else
16     img_nr2 = (nr_img/2) + 1;
17 end %if
18
19 name_nr1 = strcat('Image',num2str(img_nr1));
20 name_nr2 = strcat('Image',num2str(img_nr2));
21
22 Img1 = Images.(name_nr1);
23 Img2 = Images.(name_nr2);
24
25 diff_img = Img1 - Img2;
```

7.3 Uniformity

7.3.1 Homogen.m

```
1  % Robin Antony Birkeland Bugge
2  % Homogeneity
```

```
3  % ************************************************************************
4
5  w = warning ('off','all'); %disables warnings for large search radii
6
7  img = dicomread(img_address);
8  img = double(img);
9  info = dicominfo(img_address);
10 dim = size(img);
11
12
13 status = get(handles.SNR_check,'value');
14 if status == 1
15     img2 = double(img2);
16 end %if
17
18
19 Sens = 0.9;
20 Edge = 0.3;
21 p = 0.25;
22
23 if ~isempty(get(handles.ROI_inn,'String'))
24     p = get(handles.ROI_inn,'String');
25     p = str2double(p)/100;
26 end %if
27
28 if ~isempty(get(handles.Sens_inn,'String'))
29     Sens = get(handles.Sens_inn,'String');
30     Sens = str2double(Sens);
31 end %if
32
33 if ~isempty(get(handles.Edge_inn,'String'))
34     Edge = get(handles.Edge_inn,'String');
35     Edge = str2double(Edge);
36 end %if
37
38 %scale default radii search values based on resolution
39 s = size(img);
40 scale = s(1)/128;
41
42 r_min = scale*20;
43 r_max = scale*100;
44
45 if ~isempty(get(handles.edit_r_min,'String'))
46     r_min = get(handles.edit_r_min,'String');
47     r_min = str2double(r_min);
48 end %if
49
50 if ~isempty(get(handles.edit_r_max,'String'))
51     r_max = get(handles.edit_r_max,'String');
52     r_max = str2double(r_max);
53 end %if
54
55 [centers_bright, radii_bright] = imfindcircles(img,[r_min r_max],'
       ObjectPolarity','bright', ...
56     'Sensitivity',Sens,'EdgeThreshold',Edge);
57
58 imshow(img,'DisplayRange',[],'Parent',handles.axes1);
59 viscircles(centers_bright,radii_bright,'Parent',handles.axes1);
60
61 % find center
62 xc = centers_bright(1);
63 yc = centers_bright(2);
```

```
64
65  hold on
66  plot(xc,yc,'x','Parent',handles.axes1)
67  hold off
68
69  % Determine ROI
70  r = p * radii_bright;
71
72  hold on
73  viscircles(centers_bright,r,'Parent',handles.axes1);
74  hold off
75
76  % signal variables
77  signal_sum = 0;
78  signal_sum3 = 0;
79
80  AAD_undivided = 0;
81  signal = zeros(dim(1),dim(2));
82  signal3 = zeros(dim(1),dim(2));
83  status = get(handles.SNR_check,'value');
84
85  dim = size(img);
86  Matrix = zeros(dim(1),2);
87
88  Matrix_N = 0;
89  N2 = 0;
90  mi = 0;
91
92  for i = 1:dim(2)
93      for j = 1:dim(1)
94          L = sqrt((i - xc)^2 + (j - yc)^2);
95          if L <= r
96              Matrix(i,j) = 1;
97              Matrix_N = Matrix_N + 1;
98              mi = mi + 1;
99          else
100             Matrix(i,j) = 0;
101         end %if
102     end %for
103     N2 = N2 + (mi - 1);
104     mi = 0;
105 end %for
106
107 Matrix = double(Matrix);
108
109 img_matrix = img.*Matrix;
110 mean_signal = sum(img_matrix(:))/Matrix_N;
111
112
113 if status == 1
114     img3 = double(img3);
115
116 img3_matrix = img3.*Matrix;
117 mean_signal3 = sum(img3_matrix(:))/Matrix_N;
118
119 img2_matrix = img2.*Matrix;
120 mean2_signal = sum(img2_matrix(:))/Matrix_N;
121
122 end %if
123 N_U = 0;
124 for y = 1:dim(2)
125
```

```
126     for x = 1:dim(1)
127
128         c_s = img(x,y);
129         l = sqrt((x - xc)^2 + (y - yc)^2);
130
131         if l <= r
132         AAD_undivided = AAD_undivided + (abs(c_s - mean_signal));
133         N_U = N_U + 1;
134         end % if
135
136     end % for
137
138 end %for
139
140 N = Matrix_N - 1;
141
142 % Calculations for both SNR1 and SNR2
143 if status == 1
144
145     Sum1 = 0;
146     Sum2 = 0;
147
148     for n = 1:dim(1)
149
150
151         for m = 1:dim(2)
152
153             L = sqrt((xc - n)^2 + (yc - m)^2);
154
155             if L <= r
156
157             Sum1 = Sum1 + (img3(n,m) - mean_signal3)^2;
158
159             end %if
160         end %for
161
162         for m_1 = 1:dim(1)-1
163
164             L = sqrt((xc - n)^2 + (yc - m_1)^2);
165
166             if L <= r
167
168             Sum2 = Sum2 + (img3(n,m_1 + 1) - img3(n,m_1))^2;
169
170             end %if
171
172         end %for
173
174     end %for
175
176     mean_signal_divided = (mean_signal + mean2_signal)/2;
177
178
179     SD1 = sqrt(Sum1/(N-1));
180     noise1 = SD1/sqrt(2);
181     SNR1 = mean_signal_divided/noise1;
182
183     set(handles.SNR1_output,'String',SNR1)
184
185     SD2 = sqrt(Sum2/(2*N2));
186     noise2 = SD2/sqrt(2);
187     SNR2 = mean_signal_divided/noise2;
```

```
188
189         set(handles.SNR2_output,'String',SNR2)
190
191     end %if
192
193     NAAD = AAD_undivided/(N_U);
194
195     U = 100*(1 - (NAAD/mean_signal)); %Uniformity
196
197     % Print output
198     set(handles.U,'String', num2str(U));
199     set(handles.mean_signal,'String', num2str(mean_signal));
200     set(handles.r,'String', num2str(r));
201     set(handles.r_percent,'String', num2str(p*100));
202
203     % ********************************************************************
204
205
206     p2 = 0.5;
207
208     if ~isempty(get(handles.ROI_inn2,'String'))
209         p2 = get(handles.ROI_inn2,'String');
210         p2 = str2double(p2)/100;
211     end %if
212
213     % Determine ROI
214     r2 = p2 * radii_bright;
215
216     hold on
217     viscircles(centers_bright,r2,'Parent',handles.axes1);
218     hold off
219
220     % signal variables
221
222     AAD_undivided = 0;
223
224     dim = size(img);
225     Matrix = zeros(dim(1),2);
226
227     Matrix_N = 0;
228     N2 = 0;
229     mi = 0;
230
231     for i = 1:dim(2)
232         for j = 1:dim(1)
233             L = sqrt((i - xc)^2 + (j - yc)^2);
234             if L <= r2
235                 Matrix(i,j) = 1;
236                 Matrix_N = Matrix_N + 1;
237                 mi = mi + 1;
238             else
239                 Matrix(i,j) = 0;
240             end %if
241         end %for
242         N2 = N2 + (mi - 1);
243         mi = 0;
244     end %for
245
246     Matrix = double(Matrix);
247
248     img_matrix = img.*Matrix;
249     mean_signal2 = sum(img_matrix(:))/Matrix_N;
```

```
250
251 if status == 1
252     img3 = double(img3);
253
254 img3_matrix = img3.*Matrix;
255 mean_signal3 = sum(img3_matrix(:))/Matrix_N;
256
257 img2_matrix = img2.*Matrix;
258 mean2_signal = sum(img2_matrix(:))/Matrix_N;
259
260 end %if
261
262 N_U = 0;
263 for y = 1:dim(2)
264
265     for x = 1:dim(1)
266
267         c_s = img(x,y);
268         l = sqrt((x - xc)^2 + (y - yc)^2);
269
270         if l <= r2
271         AAD_undivided = AAD_undivided + (abs(c_s - mean_signal2));
272         N_U = N_U + 1;
273         end % if
274
275     end % for
276
277 end %for
278
279 N = Matrix_N - 1;
280
281 % Calculations for both SNR1 and SNR2
282 if status == 1
283
284     Sum1 = 0;
285     Sum2 = 0;
286
287     for n = 1:dim(1)
288
289
290         for m = 1:dim(2)
291
292             L = sqrt((xc - n)^2 + (yc - m)^2);
293
294             if L <= r2
295
296             Sum1 = Sum1 + (img3(n,m) - mean_signal3)^2;
297
298             end %if
299         end %for
300
301         for m_1 = 1:dim(1)-1
302
303             L = sqrt((xc - n)^2 + (yc - m_1)^2);
304
305             if L <= r2
306
307             Sum2 = Sum2 + (img3(n,m_1 + 1) - img3(n,m_1))^2;
308
309             end %if
310
311         end %for
```

```
312
313         end %for
314
315         mean_signal_divided = (mean_signal2 + mean2_signal)/2;
316
317         SD1 = sqrt(Sum1/(N-1));
318         noise1 = SD1/sqrt(2);
319         SNR1_2 = mean_signal_divided/noise1;
320
321         set(handles.SNR1_output2,'String',SNR1_2)
322
323         SD2 = sqrt(Sum2/(2*N2));
324         noise2 = SD2/sqrt(2);
325         SNR2_2 = mean_signal_divided/noise2;
326
327         set(handles.SNR2_output2,'String',SNR2_2)
328
329     end %if
330
331     NAAD = AAD_undivided/(N_U);
332
333     U_50 = 100*(1 - (NAAD/mean_signal2));
334
335     % Print output
336     set(handles.U2,'String', num2str(U_50));
337     set(handles.mean_signal2,'String', num2str(mean_signal2));
338     set(handles.r2,'String', num2str(r2));
339     set(handles.r_percent2,'String', num2str(p2*100));
340
341     % ************************************************************************
342
343     p3 = 0.85;
344
345     if ~isempty(get(handles.ROI_inn3,'String'))
346         p3 = get(handles.ROI_inn3,'String');
347         p3 = str2double(p3)/100;
348     end %if
349
350     % Determine ROI
351     r3 = p3 * radii_bright;
352
353     hold on
354     viscircles(centers_bright,r3,'Parent',handles.axes1);
355     hold off
356
357     % signal variables
358
359     AAD_undivided = 0;
360
361     dim = size(img);
362     Matrix = zeros(dim(1),2);
363
364     Matrix_N = 0;
365     N2 = 0;
366     mi = 0;
367
368     for i = 1:dim(2)
369         for j = 1:dim(1)
370             L = sqrt((i - xc)^2 + (j - yc)^2);
371             if L <= r3
372                 Matrix(i,j) = 1;
373                 Matrix_N = Matrix_N + 1;
```

```
374                 mi = mi + 1;
375             else
376                 Matrix(i,j) = 0;
377             end %if
378         end %for
379         N2 = N2 + (mi - 1);
380         mi = 0;
381     end %for
382
383     Matrix = double(Matrix);
384
385     img_matrix = img.*Matrix;
386     mean_signal_3 = sum(img_matrix(:))/Matrix_N;
387
388     if status == 1
389         img3 = double(img3);
390
391     img3_matrix = img3.*Matrix;
392     mean_signal3 = sum(img3_matrix(:))/Matrix_N;
393
394     img2_matrix = img2.*Matrix;
395     mean2_signal = sum(img2_matrix(:))/Matrix_N;
396
397     end %if
398
399     N_U = 0;
400     for y = 1:dim(2)
401
402         for x = 1:dim(1)
403
404             c_s = img(x,y);
405             l = sqrt((x - xc)^2 + (y - yc)^2);
406
407             if l <= r3
408             AAD_undivided = AAD_undivided + (abs(c_s - mean_signal_3));
409             N_U = N_U + 1;
410             end % if
411
412         end % for
413
414     end %for
415
416     N = Matrix_N - 1;
417
418     % Calculations for both SNR1 and SNR2
419     if status == 1
420
421         Sum1 = 0;
422         Sum2 = 0;
423
424         for n = 1:dim(1)
425
426
427             for m = 1:dim(2)
428
429                 L = sqrt((xc - n)^2 + (yc - m)^2);
430
431                 if L <= r3
432
433                 Sum1 = Sum1 + (img3(n,m) - mean_signal3)^2;
434
435                 end %if
```

```
436             end %for
437
438             for m_1 = 1:dim(1)-1
439
440                 L = sqrt((xc - n)^2 + (yc - m_1)^2);
441
442                 if L <= r3
443
444                 Sum2 = Sum2 + (img3(n,m_1 + 1) - img3(n,m_1))^2;
445
446                 end %if
447
448             end %for
449
450         end %for
451
452         mean_signal_divided = (mean_signal_3 + mean2_signal)/2;
453
454
455         SD1 = sqrt(Sum1/(N-1));
456         noise1 = SD1/sqrt(2);
457         SNR1_3 = mean_signal_divided/noise1;
458
459         set(handles.SNR1_output3,'String',SNR1_3)
460
461         SD2 = sqrt(Sum2/(2*N2));
462         noise2 = SD2/sqrt(2);
463         SNR2_3 = mean_signal_divided/noise2;
464
465         set(handles.SNR2_output3,'String',SNR2_3)
466     end %if
467
468     NAAD = AAD_undivided/(N_U);
469
470     U_85 = 100*(1 - (NAAD/mean_signal_3));
471
472     % Print output
473     set(handles.U3,'String', num2str(U_85));
474     set(handles.mean_signal3,'String', num2str(mean_signal_3));
475     set(handles.r3,'String', num2str(r3));
476     set(handles.r_percent3,'String', num2str(p3*100));
477
478     % ************************************************************************
479
480
481     %Printer
482
483     Tagprinter_homogen
484     Infoprinter_homogen
```

7.3.2 Homogen_reader.m

```
1   % Robin Antony Birkeland Bugge
2   % Homogeneity reader
3   % ************************************************************************
4
5
6   [Filename,PathName,FilterIndex] = uigetfile('*.*');
7   img_address = strcat(PathName,Filename);
8   img = dicomread(img_address);
```

```
img = double(img);

status = get(handles.SNR_check,'value');
if status == 1
    [Filename2,PathName2,FilterIndex2] = uigetfile('*.*');
    img_address2 = strcat(PathName2,Filename2);
    img2 = dicomread(img_address2);
    img2 = double(img2);
end %if
```

7.3.3 Infoprinter_homogen

```
% Robin Antony Birkeland Bugge. November 2013
% info_printer for homogeneity module
% ************************************************************************


% Spacer
fprintf(fid,'\n');

if ~isempty(get(handles.ROI_inn,'String'))

    ROI_s1 = get(handles.ROI_inn,'String');

else
    ROI_s1 = 25;

end %if

if ~isempty(get(handles.ROI_inn2,'String'))

    ROI_s2 = get(handles.ROI_inn2,'String');

else
    ROI_s2 = 50;

end %if

if ~isempty(get(handles.ROI_inn3,'String'))

    ROI_s3 = get(handles.ROI_inn3,'String');

else
    ROI_s3 = 85;

end %if

% Print header for info
fprintf(fid,'%s ','ROI size:');
fprintf(fid,'%2.f,',ROI_s1);
fprintf(fid,'%2.f,',ROI_s2);
fprintf(fid,'%2.f, \n',ROI_s3);

if status == 1

S = {'Uniformity','Mean signal','ROI radii (pxl)','ROI radii (%)','SNR1','
    SNR2'};

D = [U,U_50,U_85; ...
     mean_signal,mean_signal2,mean_signal_3; ...
```

```
48          r,r2,r3; ...
49          p*100,p2*100,p3*100; ...
50          SNR1, SNR1_2,SNR1_3; ...
51          SNR2, SNR2_2, SNR2_3];
52  else
53
54
55  S = {'Uniformity','Mean signal','ROI radii (pxl)','ROI radii (%)'};
56
57  D = [U,U_50,U_85; ...
58          mean_signal,mean_signal2,mean_signal_3; ...
59          r,r2,r3; ...
60          p*100,p2*100,p3*100];
61
62  end %if
63
64  % Prints the matrix D that contains info on each object
65  for i = 1:size(D,1)
66      fprintf(fid,'%s, ',S{i});
67      fprintf(fid, '%f,', D(i,:));
68      fprintf(fid,'\n');
69  end %for
```

7.3.4 Tagprinter_homogen.m

```
1   % Robin Antony Birkeland Bugge. November 2013
2   % Tag printer
3   % ************************************************************************
4
5   [FileName,Pathname] = uiputfile('*.xls');
6   info = dicominfo(img_address);
7   file = strcat(Pathname,'\',FileName);
8   fid = fopen(file,'wt');
9
10  fprintf(fid,'%s\n',' ');
11
12  if isfield(info,'ImagePositionPatient') == 1
13
14      Tags.ImagePositionPatient = info.ImagePositionPatient;
15      Field = 'ImagePositionPatient';
16
17      fprintf(fid,'%s:',Field);
18      fprintf(fid,'%6.2f,%6.2f,%6.2f\n',Tags.ImagePositionPatient);
19
20  end %if
21
22  if isfield(info,'SeriesDescription') == 1
23
24      Tags.SeriesDescription = info.SeriesDescription;
25      Field = 'SeriesDescription';
26
27      fprintf(fid,'%s:',Field);
28      fprintf(fid,'%s\n',Tags.SeriesDescription);
29
30  end %if
31
32  if isfield(info,'ImageOrientationPatient') == 1
33      Tags.ImageOrientationPatient = info.ImageOrientationPatient;
34      Field = 'ImageOrientationPatient';
35
```

```
36      if info.ImageOrientationPatient(1) == 1 && abs(info.
            ImageOrientationPatient(6)) == 1
37          % Kor
38          fprintf(fid,'%s:',Field);
39          fprintf(fid,'%s\n','Kor');
40      end
41
42      if info.ImageOrientationPatient(1) == 1 && abs(info.
            ImageOrientationPatient(5)) == 1
43          % Tra
44          fprintf(fid,'%s:',Field);
45          fprintf(fid,'%s\n','Tra');
46      end
47
48      if info.ImageOrientationPatient(2) == 1 && abs(info.
            ImageOrientationPatient(6)) == 1
49          % Sag
50          fprintf(fid,'%s:',Field);
51          fprintf(fid,'%s\n','Sag');
52      end
53
54  end %if
55
56  if isfield(info,'Rows') == 1
57      Tags.Rows = info.Rows;
58      Field = 'Rows';
59
60      fprintf(fid,'%s:',Field);
61      fprintf(fid,'%6.2f\n',Tags.Rows);
62
63      if isfield(info,'PixelSpacing') == 1
64          fprintf(fid,'%s:','FieldOfView_Rows');
65          FieldOfView_row = Tags.Rows * info.PixelSpacing(1);
66          fprintf(fid,'%6.2f\n',FieldOfView_row);
67      end %if
68
69
70  end %if
71
72  if isfield(info,'Columns') == 1
73      Tags.Columns = info.Columns;
74      Field = 'Columns';
75
76      fprintf(fid,'%s:',Field);
77      fprintf(fid,'%6.2f\n',Tags.Columns);
78
79      if isfield(info,'PixelSpacing') == 1
80          fprintf(fid,'%s:','FieldOfView_Columns');
81          FieldOfView_column = Tags.Columns * info.PixelSpacing(1);
82          fprintf(fid,'%6.2f\n',FieldOfView_column);
83      end %if
84
85  end %if
86
87  if isfield(info,'PixelSpacing') == 1
88      Tags.PixelSpacing = info.PixelSpacing;
89      Field = 'PixelSpacing';
90
91      fprintf(fid,'%s:',Field);
92      fprintf(fid,'%6.2f,%6.2f\n',Tags.PixelSpacing);
93
94  end %if
```

```
95
96  if isfield(info,'AcquisitionDate') == 1
97      Tags.AcquisitionDate = info.AcquisitionDate;
98      Field = 'AcquisitionDate';
99
100     fprintf(fid,'%s:',Field);
101     fprintf(fid,'%s\n',Tags.AcquisitionDate);
102
103 end %if
104
105 if isfield(info,'AcquisitionTime') == 1
106     Tags.AcquisitionTime = info.AcquisitionTime;
107     Field = 'AcquisitionTime';
108
109     fprintf(fid,'%s:',Field);
110     fprintf(fid,'%s\n',Tags.AcquisitionTime);
111
112 end %if
113
114 if isfield(info,'Manufacturer') == 1
115     Tags.Manufacturer = info.Manufacturer;
116     Field = 'Manufacturer';
117
118     fprintf(fid,'%s:',Field);
119     fprintf(fid,'%s\n',Tags.Manufacturer);
120
121 end %if
122
123 if isfield(info,'InstitutionName') == 1
124     Tags.InstitutionName = info.InstitutionName;
125     Field = 'InstitutionName';
126
127     fprintf(fid,'%s:',Field);
128     fprintf(fid,'%s\n',Tags.InstitutionName);
129
130 end %if
131
132 if isfield(info,'StationName') == 1
133     Tags.StationName = info.StationName;
134     Field = 'StationName';
135
136     fprintf(fid,'%s:',Field);
137     fprintf(fid,'%s\n',Tags.StationName);
138
139 end %if
140
141 if isfield(info,'SliceThickness') == 1
142     Tags.SliceThickness = info.SliceThickness;
143     Field = 'SliceThickness';
144
145     fprintf(fid,'%s:',Field);
146     fprintf(fid,'%6.2f\n',Tags.SliceThickness);
147
148 end %if
149
150 seperator = '
    ********************************************************************************
    ';
151 fprintf(fid,'%s\n','');
152 fprintf(fid,'%s:',seperator);
153
154 % ************************************************************************
```

```
155 % Second image
156 if status == 1
157
158 address2 = get(handles.address_line2,'String');
159
160 info = dicominfo(address2);
161
162 fprintf(fid,'%s\n',' ');
163
164 if isfield(info,'ImagePositionPatient') == 1
165
166     Tags.ImagePositionPatient = info.ImagePositionPatient;
167     Field = 'ImagePositionPatient';
168
169     fprintf(fid,'%s:',Field);
170     fprintf(fid,'%6.2f,%6.2f,%6.2f\n',Tags.ImagePositionPatient);
171
172 end %if
173
174 if isfield(info,'SeriesDescription') == 1
175
176     Tags.SeriesDescription = info.SeriesDescription;
177     Field = 'SeriesDescription';
178
179     fprintf(fid,'%s:',Field);
180     fprintf(fid,'%s\n',Tags.SeriesDescription);
181
182 end %if
183
184 if isfield(info,'ImageOrientationPatient') == 1
185     Tags.ImageOrientationPatient = info.ImageOrientationPatient;
186     Field = 'ImageOrientationPatient';
187
188     if info.ImageOrientationPatient(1) == 1 && abs(info.
          ImageOrientationPatient(6)) == 1
189         % Kor
190         fprintf(fid,'%s:',Field);
191         fprintf(fid,'%s\n','Kor');
192     end
193
194     if info.ImageOrientationPatient(1) == 1 && abs(info.
          ImageOrientationPatient(5)) == 1
195         % Tra
196         fprintf(fid,'%s:',Field);
197         fprintf(fid,'%s\n','Tra');
198     end
199
200     if info.ImageOrientationPatient(2) == 1 && abs(info.
          ImageOrientationPatient(6)) == 1
201         % Sag
202         fprintf(fid,'%s:',Field);
203         fprintf(fid,'%s\n','Sag');
204     end
205
206 end %if
207
208 if isfield(info,'Rows') == 1
209     Tags.Rows = info.Rows;
210     Field = 'Rows';
211
212     fprintf(fid,'%s:',Field);
213     fprintf(fid,'%6.2f\n',Tags.Rows);
```

```
214
215     if isfield(info,'PixelSpacing') == 1
216         fprintf(fid,'%s:','FieldOfView_Rows');
217         FieldOfView_row = Tags.Rows * info.PixelSpacing(1);
218         fprintf(fid,'%6.2f\n',FieldOfView_row);
219     end %if
220
221
222 end %if
223
224 if isfield(info,'Columns') == 1
225     Tags.Columns = info.Columns;
226     Field = 'Columns';
227
228     fprintf(fid,'%s:',Field);
229     fprintf(fid,'%6.2f\n',Tags.Columns);
230
231     if isfield(info,'PixelSpacing') == 1
232         fprintf(fid,'%s:','FieldOfView_Columns');
233         FieldOfView_column = Tags.Columns * info.PixelSpacing(1);
234         fprintf(fid,'%6.2f\n',FieldOfView_column);
235     end %if
236
237 end %if
238
239 if isfield(info,'PixelSpacing') == 1
240     Tags.PixelSpacing = info.PixelSpacing;
241     Field = 'PixelSpacing';
242
243     fprintf(fid,'%s:',Field);
244     fprintf(fid,'%6.2f,%6.2f\n',Tags.PixelSpacing);
245
246 end %if
247
248 if isfield(info,'AcquisitionDate') == 1
249     Tags.AcquisitionDate = info.AcquisitionDate;
250     Field = 'AcquisitionDate';
251
252     fprintf(fid,'%s:',Field);
253     fprintf(fid,'%s\n',Tags.AcquisitionDate);
254
255 end %if
256
257 if isfield(info,'AcquisitionTime') == 1
258     Tags.AcquisitionTime = info.AcquisitionTime;
259     Field = 'AcquisitionTime';
260
261     fprintf(fid,'%s:',Field);
262     fprintf(fid,'%s\n',Tags.AcquisitionTime);
263
264 end %if
265
266 if isfield(info,'Manufacturer') == 1
267     Tags.Manufacturer = info.Manufacturer;
268     Field = 'Manufacturer';
269
270     fprintf(fid,'%s:',Field);
271     fprintf(fid,'%s\n',Tags.Manufacturer);
272
273 end %if
274
275 if isfield(info,'InstitutionName') == 1
```

```
276         Tags.InstitutionName = info.InstitutionName;
277         Field = 'InstitutionName';
278 
279         fprintf(fid,'%s:',Field);
280         fprintf(fid,'%s\n',Tags.InstitutionName);
281 
282     end %if
283 
284     if isfield(info,'StationName') == 1
285         Tags.StationName = info.StationName;
286         Field = 'StationName';
287 
288         fprintf(fid,'%s:',Field);
289         fprintf(fid,'%s\n',Tags.StationName);
290 
291     end %if
292 
293     if isfield(info,'SliceThickness') == 1
294         Tags.SliceThickness = info.SliceThickness;
295         Field = 'SliceThickness';
296 
297         fprintf(fid,'%s:',Field);
298         fprintf(fid,'%6.2f\n',Tags.SliceThickness);
299 
300     end %if
301 
302     seperator = '
        ********************************************************************************
        ';
303     fprintf(fid,'%s\n','');
304     fprintf(fid,'%s:',seperator);
305     end %if
```

7.4 Artificial MR image generator

7.4.1 Artificer.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Artificial test image artificer
3 
4  % ************************************************************************
5 
6  clear all
7 
8  % Lager philips fantomet
9  % ************************************************************************
10 
11 %Load image for size determination
12 size_img = dicomread('Size_comp');
13 s = 1;
14 dim = s*size(size_img);
15 r = s*8;
16 R = s*200;
17 
18 xc = dim(2)/2;
19 yc = dim(1)/2;
20 
21 Image = zeros(dim(1),dim(2));
```

```
22  Image = mat2gray(Image);
23
24
25  philips_artificer
26
27  %warp_artificer
28
29  l = size(positions);
30
31  figure(1)
32  for i = 1:l(1)
33
34  for x = 1:dim(2)
35
36      for y = 1:dim(1)
37
38          L = sqrt((x-positions(i,1))^2 + (y-positions(i,2))^2);
39          Lc = sqrt((x-xc)^2 + (y-yc)^2);
40
41          if L <= r
42
43              Image(x,y) = 0.5;
44
45          end %if
46
47          if Lc <= R
48
49              Image(x,y) = Image(x,y) + 0.001;
50
51          end %if
52
53       end %for
54
55
56  end %for
57
58  end %for
59
60  % ************************************************************************
61
62
63  % Lager homogent fantom, kan ikke kjøres samtidig som philips makeren
64  % ************************************************************************
65
66  % Load image for size determination
67  % size_img = dicomread('Size_comp_homogen');
68  % dim = size(size_img);
69  % r = 8;
70  % R = 40;
71  %
72  % xc = dim(2)/2;
73  % yc = dim(1)/2;
74  %
75  % name = 'Homogen';
76  %
77  % Image = zeros(dim(1),dim(2));
78  % Image = mat2gray(Image);
79  %
80  % figure(1)
81  %
82  % for x = 1:dim(2)
83  %
```

```
%      for y = 1:dim(1)
%
%          Lc = sqrt((x-xc)^2 + (y-yc)^2);
%
%          if Lc <= R
%
%              Image(x,y) = Image(x,y) + 0.4;
%
%          end %if
%
%       end %for
%
%
%
% end %for


% ************************************************************************

name = 'Fasit_num';

% Create random noise. Rician distribution

noice_vec = 0.05*randn(dim(1),dim(2));
%Image = Image + noice_vec;


imshow(Image)
dicomwrite(Image,strcat('C:\Robin\Fysikk\Masteroppgaven\Programvare\Beta\
    Artificial Images\Bilder\',name));
```

7.4.2 Artificer_drift.m

```
% Robin Antony Birkeland Bugge. October 2013
% Artificial test image artificer

% ************************************************************************

clear all

N = 10;

Standard = 0.001;
max_drift = 0.001;
drift_vec = linspace(0,max_drift,N);

o = 1;

while o <= N

number = num2str(o);
name = strcat('Image',number);

% Lager philips fantomet
% ************************************************************************

%Load image for size determination

s = 1;
dim = s*[448 448];
```

```
r = s*5;
R = s*200;

xc = dim(2)/2;
yc = dim(1)/2;

Image = zeros(dim(1),dim(2));
Image = mat2gray(Image);


l = size(Image);

for i = 1:l(1)

for x = 1:dim(2)

    for y = 1:dim(1)

        Lc = sqrt((x-xc)^2 + (y-yc)^2);

        if Lc <= R

            Image(x,y) = Image(x,y) + 0.001;
            Image(x,y) = Image(x,y) + drift_vec(o);

        end %if

     end %for


end %for

end %for

% ************************************************************************

% Create random noise. Rician distribution

% noice_vec = 0.05*randn(dim(1),dim(2));
% Image = Image + noice_vec;

figure(o)
imshow(Image)
dicomwrite(Image,strcat('C:\Robin\Fysikk\Masteroppgaven\Tester\
    Auto_vs_manuell\Drift_test\Bilder\',name,'.dcm'));

o = o+1;

end %while
```

7.4.3 Artificer_repeat.m

```
% Robin Antony Birkeland Bugge. October 2013
% Artificial test image artificer

% ************************************************************************

clear all

N = 1;
```

```
p1_vec = 1*randn(N,1);
p2_vec = 1*randn(N,1);
p1_y_vec = 1*randn(N,1);
p2_y_vec = 1*randn(N,1);

o = 1;

while o <= N

% number = num2str(o);
% name = strcat('Bilde',number);
name = 'Bilde17';
% Lager philips fantomet
% ************************************************************************

%Load image for size determination
size_img = dicomread('Size_comp');
s = 1;
dim = s*size(size_img);
r = s*5;
R = s*200;

xc = dim(2)/2;
yc = dim(1)/2;

Image = zeros(dim(1),dim(2));
Image = mat2gray(Image);


philips_artificer

warp_artificer_rand



h = 1;

l = size(positions);

for i = 1:l(1)

for x = 1:dim(2)

    for y = 1:dim(1)

        L = sqrt((x-positions(i,1))^2 + (y-positions(i,2))^2);
        Lc = sqrt((x-xc)^2 + (y-yc)^2);



        if L <= r

            Image(x,y) = 0.6;

        end %if

        if Lc <= R

            Image(x,y) = Image(x,y) + 0.001;

        end %if

```

```
71        end %for
72
73
74  end %for
75
76  end %for
77
78  % ************************************************************************
79
80
81  % Lager homogent fantom, kan ikke kjøres samtidig som philips makeren
82  % ************************************************************************
83
84  % Load image for size determination
85  % size_img = dicomread('Size_comp_homogen');
86  % dim = size(size_img);
87  % r = 8;
88  % R = 40;
89  %
90  % xc = dim(2)/2;
91  % yc = dim(1)/2;
92  %
93  % name = 'Homogen';
94  %
95  % Image = zeros(dim(1),dim(2));
96  % Image = mat2gray(Image);
97  %
98  % figure(1)
99  %
100 % for x = 1:dim(2)
101 %
102 %     for y = 1:dim(1)
103 %
104 %         Lc = sqrt((x-xc)^2 + (y-yc)^2);
105 %
106 %         if Lc <= R
107 %
108 %             Image(x,y) = Image(x,y) + 0.4;
109 %
110 %         end %if
111 %
112 %      end %for
113 %
114 %
115 %
116 % end %for
117
118
119 % ************************************************************************
120
121 % Create random noise. Rician distribution
122
123 noice_vec = 0.05*randn(dim(1),dim(2));
124 Image = Image + noice_vec;
125
126 figure(o)
127 imshow(Image)
128 dicomwrite(Image,strcat('C:\Robin\Fysikk\Masteroppgaven\Tester\
        Auto_vs_manuell\Repeat_test\Bilder\',name));
129
130 o = o+1;
131
```

```
132 end %while
```

7.4.4 Easy_reader.m

```
1  img_name = 'C:\Robin\Fysikk\Masteroppgaven\Tester\Auto_vs_manuell\
       Repeat_test\Bilder\Bilde100';
2
3  img = dicomread(img_name);
4
5  imshow(img,'Displayrange',[])
```

7.4.5 philips_artificer.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Vector with philips phantom positions.
3
4  % ********************************************************************
5
6  % positions = [xc - 3*50 yc - 2*50; xc - 3*50 yc - 1*50; xc - 3*50 yc; xc -
       3*50 yc + 50; xc - 3*50 yc + 2*50; ...
7  %              xc - 2*50 yc - 3*50; xc - 2*50 yc - 2*50; xc - 2*50 yc -
       1*50; xc - 2*50 yc; xc - 2*50 yc + 50; xc - 2*50 yc + 2*50; xc - 2*50 yc
        + 3*50; ...
8  %              xc - 1*50 yc - 3*50; xc - 1*50 yc - 2*50; xc - 1*50 yc -
       1*50; xc - 1*50 yc; xc - 1*50 yc + 50; xc - 1*50 yc + 2*50; xc - 1*50 yc
        + 3*50; ...
9  %              xc        yc - 3*50; xc        yc - 2*50; xc        yc -
       1*50; xc        yc; xc        yc + 50; xc        yc + 2*50; xc        yc
        + 3*50; ...
10 %              xc + 1*50 yc - 3*50; xc + 1*50 yc - 2*50; xc + 1*50 yc -
       1*50; xc + 1*50 yc; xc + 1*50 yc + 50; xc + 1*50 yc + 2*50; xc + 1*50 yc
        + 3*50; ...
11 %              xc + 2*50 yc - 3*50; xc + 2*50 yc - 2*50; xc + 2*50 yc -
       1*50; xc + 2*50 yc; xc + 2*50 yc + 50; xc + 2*50 yc + 2*50; xc + 2*50 yc
        + 3*50; ...
12 %              xc + 3*50 yc - 2*50; xc + 3*50 yc - 1*50; xc + 3*50 yc; xc +
       3*50 yc + 50; xc + 3*50 yc + 2*50];
13
14
15
16 % Larger positions
17 positions = [xc - 3*50*s yc - 2*50*s; xc - 3*50*s yc - 1*50*s; xc - 3*50*s
       yc; xc - 3*50*s yc + 50*s; xc - 3*50*s yc + 2*50*s; ...
18               xc - 2*50*s yc - 3*50*s; xc - 2*50*s yc - 2*50*s; xc - 2*50*s
                     yc - 1*50*s; xc - 2*50*s yc; xc - 2*50*s yc + 50*s; xc -
                     2*50*s yc + 2*50*s; xc - 2*50*s yc + 3*50*s; ...
19               xc - 1*50*s yc - 3*50*s; xc - 1*50*s yc - 2*50*s; xc - 1*50*s
                     yc - 1*50*s; xc - 1*50*s yc; xc - 1*50*s yc + 50*s; xc -
                     1*50*s yc + 2*50*s; xc - 1*50*s yc + 3*50*s; ...
20               xc        yc - 3*50*s; xc        yc - 2*50*s; xc        yc -
                     1*50*s; xc        yc; xc        yc + 50*s; xc        yc +
                     2*50*s; xc        yc + 3*50*s; ...
21               xc + 1*50*s yc - 3*50*s; xc + 1*50*s yc - 2*50*s; xc + 1*50*s
                     yc - 1*50*s; xc + 1*50*s yc; xc + 1*50*s yc + 50*s; xc +
                     1*50*s yc + 2*50*s; xc + 1*50*s yc + 3*50*s; ...
22               xc + 2*50*s yc - 3*50*s; xc + 2*50*s yc - 2*50*s; xc + 2*50*s
                     yc - 1*50*s; xc + 2*50*s yc; xc + 2*50*s yc + 50*s; xc +
                     2*50*s yc + 2*50*s; xc + 2*50*s yc + 3*50*s; ...
```

```
23              xc + 3*50*s yc - 2*50*s; xc + 3*50*s yc - 1*50*s; xc + 3*50*s
                    yc; xc + 3*50*s yc + 50*s; xc + 3*50*s yc + 2*50*s];
```

7.4.6 warp_artificer.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Warp artificer
3
4  % ************************************************************************
5
6  %linear warping
7
8  p1 = 2;
9  p2 = 2;
10
11 p1_y = 3;
12 p2_y = 1.3;
13
14 dim_pos = size(positions);
15
16 positions_old = positions;
17
18 second_y = zeros(dim_pos(1),1);
19 second_x = zeros(dim_pos(2),1);
20
21 for i = 1:dim_pos(1)
22     second_y(i) = ((positions(i,1)-yc)*(p2/100))^2;
23     if positions(i,1)-yc < 0
24         second_y(i) = -second_y(i);
25     end
26
27     second_x(i) = ((positions(i,2)-xc)*(p2_y/100))^2;
28     if positions(i,2)-xc < 0
29         second_x(i) = -second_x(i);
30     end
31
32 end
33
34
35 positions_y = positions(:,1) + (positions(:,1)-yc)*(p1/100) + second_y;
36 positions_x = positions(:,2) + (positions(:,2)-xc)*(p1_y/100) + second_x;
37 positions = [positions_y, positions(:,2)]; %yretning
38 positions = [positions(:,1), positions_x]; %xretning
39
40
41
42 lengths_old = sqrt((positions_old(:,1) - xc).^2 + (positions_old(:,2)-yc)
       .^2);
43
44 lengths = sqrt((positions(:,1) - xc).^2 + (positions(:,2)-yc).^2);
45
46
47 difference = lengths - lengths_old;
48 diff_per = [lengths_old,lengths,difference,((difference./lengths_old)*100)];
```

7.4.7 warp_artificer_rand.m

```
1  % Robin Antony Birkeland Bugge. October 2013
2  % Warp artificer
```

```
3
4  % ************************************************************************
5
6  %linear warping
7
8  p1 = p1_vec(o);
9  p2 = p2_vec(o);
10
11 p1_y = p1_y_vec(o);
12 p2_y = p2_y_vec(o);
13
14 dim_pos = size(positions);
15
16 positions_old(:,:) = positions(:,:);
17
18 second_y = zeros(dim_pos(1),1);
19 second_x = zeros(dim_pos(2),1);
20
21 for i = 1:dim_pos(1)
22     second_y(i) = ((positions(i,1)-yc)*(p2/100))^2;
23     if positions(i,1)-yc < 0
24         second_y(i) = -second_y(i);
25     end
26
27     second_x(i) = ((positions(i,2)-xc)*(p2_y/100))^2;
28     if positions(i,2)-xc < 0
29         second_x(i) = -second_x(i);
30     end
31
32 end
33
34 positions_x = positions(:,1) + (positions(:,1)-yc)*(p1/100) + second_y;
35 positions_y = positions(:,2) + (positions(:,2)-xc)*(p1_y/100) + second_x;
36 positions = [positions_y, positions(:,2)]; %yretning
37 positions = [positions(:,1), positions_x]; %xretning
38
39
40 lengths_old = sqrt((positions_old(:,1) - xc).^2 + (positions_old(:,2)-yc)
       .^2);
41 lengths = sqrt((positions(:,1) - xc).^2 + (positions(:,2)-yc).^2);
42 differance = lengths - lengths_old;
43
44 Current_fasit = [positions_x,positions_y,lengths_old,lengths,differance,((
       differance./lengths_old)*100)];
45 Fasit(:,:,o) = Current_fasit;
46 Distortion(:,:,o) = ((differance./lengths_old)*100);
```

Bibliography

[1] National Electrical Manufacturers Association, *NEMA Standards Publication MS 2-2003: Determination of Two-Dimensional Geometric Distortion in Diagnostic Magnetic Resonance Images*, Electronic Article, 2003.

[2] National Electrical Manufacturers Association, *NEMA Standards Publication MS 5-2003: Determination of Slice Thickness in Diagnostic Magnetic Resonance Imaging*, Electronic Article, 2003

[3] National Electrical Manufacturers Association, *NEMA Standards Publication MS 1-2008: Determination of Signal-to-Noise Ratio (SNR) in Diagnostic Magnetic Resonance Imaging*, Electronic Article, 2008.

[4] National Electrical Manufacturers Association, *NEMA Standards Publication MS 3-2008: Determination of Image Uniformity in Diagnostic Magnetic Resonance Images*, 2008.

[5] Internation Electrotechnical Commission, *International Standard, Magnetic resonance equipment for medical imaging*, Electronic Article, 2007.

[6] Bjørnerud, Atle, *The Physics of Magnetic Resonance Imaging: FYS-KJM 4740*, Book, 2008.

[7] Friedman, L. and Glover, G. H, *Report on a multicenter fMRI quality assurance protocol*, Journal of magnetic resonance imaging : JMRI, volume 23, Review,Research Support, N.I.H, Extramural, `http://europepmc.org/abstract/MED/16649196`, Journal Article, 2006.

[8] Vlaardingerbroek, M.T. and Boer, J.A, *Magnetic Resonance Imaging: Theory and Practice*, Springer, Book, `http://books.google.no/books?id=rcwv_puzLSQC`, 2003.

[9] Robert M. Weisskoff, *Simple Measurement of Scanner Stability for Functional NMR Imaging of Activation in the Brain*, `http://www.ncbi.nlm.nih.gov/pubmed/8892220`, Journal Article, 1996.

[10] Centre for Evidence-based Purchasing, *Report 06006, 3T MRI systems*, Magnet comparison report, Issue 4, 2007.

[11] Centre for Evidence-based Purchasing, *Report 05047, 1.5T MRI systems*, Magnet comparison report, Issue 5, 2005.

[12] Friberg EG, Widmark A, Olerud HM, Tynes T, Saxebøl G, *Guidance for use of medical X-ray and MR equipment subjected to approval. Guidance to Regulations for radiation protection and use of radiation. Guidance No. 5 Østerås: Norwegian Radiation Protection Authority*, 2005.

[13] MathWorks, *imfindcircles. Find circles using circular Hough transform*, Company internet page, `http://www.mathworks.se/help/images/ref/imfindcircles.html`, 2014.

[14] Zhiyue J. Wang, Youngseob Seo, Jonathan M. Chia, Nancy K. Rollins, *A quality assurance protocol for diffusion tensor imaging using the head phantom from American College of Radiology*, Article, 2011.

[15] Bodurka, J., and P. Bandettini, *Simple Model to Describe Weisskoff EPI Temporal Stability Test-Analogy with Physiological Noise Model in Oxygenation-Sensitive fMRI*, Proc. Intl. Soc. Mag. Reson. Med. Vol. 14, 2006.

[16] Macovski. A, *Noise in MRI*, Magn Reson Med. 36, 1996.

[17] Zhu, Hongtu, et al, *Regression models for identifying noise sources in magnetic resonance images.*, Journal of the American Statistical Association 104.486: 623-637, 2009.

[18] Rafael C. Gonzalez, Richard E. Woods, *Digital Image Processing, Third Edition*, Pearson Prentice Hall, 2008.

[19] Tone Elise Døli Orheim, Wibeke Nordhøy, *Book of methods for reception control of MR scanners*, Ullevål Hospital, Section for diagnostic physics, Oslo, Norway, August 2010.

[20] Rasband, W.S., U. S. National Institutes of Health. ImageJ. Bethesda, Maryland, USA, `http://imagej.nih.gov/ij/`, 1997-2011.